### Menus and Toolbars Overview

The [C1Command](#_bookmark243) 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](#_bookmark243) are [C1MainMenu,](#_bookmark773) [C1CommandMenu,](#_bookmark500) [C1CommandControl,](#_bookmark306) [C1CommandMdiList,](#_bookmark483) and [C1ContextMenu.](#_bookmark576) Whereas, the main types of objects for building toolbar systems are [C1ToolBar,](#_bookmark1170) [C1CommandMenu,](#_bookmark500) [C1CommandControl,](#_bookmark306) [C1CommandMdiList,](#_bookmark483) and [C1ContextMenu.](#_bookmark576) The only difference between the two is the two primary controls: [C1MainMenu](#_bookmark773) and [C1ToolBar.](#_bookmark1170)

**C1MainMenu**

[C1MainMenu](#_bookmark773) 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](#_bookmark350) will automatically appear in the component tray. The [C1CommandHolder](#_bookmark350) stores all of the menu's commands as a single collection. For more information on how to use the [C1CommandHolder,](#_bookmark350) please see [C1CommandHolder Component.](#_bookmark51)

Command links of type [C1CommandLink](#_bookmark416) are used to represent the commands in menus.

**C1ToolBar**

**C1ToolBar** is a control which represents a toolbar. Like the [C1MainMenu](#_bookmark773) it contains a collection of command links stored in the [C1CommandHolder](#_bookmark350) 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.

**See Also**

[Menus and Toolbars Functionality](#_bookmark49)

[Common Objects Used to Create Menus and Toolbars](#_bookmark50) [Unique Objects Among Menus and Toolbars](#_bookmark56)

[Menus Appearance and Behavior](#_bookmark59)

[Toolbars Appearance and Behavior](#_bookmark67)

#### 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](#_bookmark243) or derived types, see [Class Hierarchy](#_bookmark11) 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 [C1Command](#_bookmark243)s menus and toolbars. Instead, all commands on a form are stored as a single collection in a component of type [C1CommandHolder,](#_bookmark350) a single instance of which is automatically created on the form when you add the first [C1Command](#_bookmark243)s menu or toolbar to it.

For more information on how to use **C1CommandHolder**, please see [C1CommandHolder](#_bookmark51) [Component.](#_bookmark51) To represent commands in menus and toolbars, **command links** (components of type [C1CommandLink)](#_bookmark416) are used.

**Functionality of a C1CommandLink Component**

A **command link** is a small and quite simple component. Its most important property is [Command,](#_bookmark430) 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](#_bookmark243) and derived classes) are automatically stored in the form’s command holder (object of type [C1CommandHolder](#_bookmark350)).
  + Menus and toolbars (objects of type [C1MainMenu,](#_bookmark773) [C1CommandMenu,](#_bookmark500) [C1ContextMenu,](#_bookmark576) [C1ToolBar)](#_bookmark1170) contain command links (type [C1CommandLink](#_bookmark416)) 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.

**See Also**

[C1CommandHolder Smart Tag](#_bookmark17) [C1CommandMenu Command](#_bookmark52) [C1ContextMenu Control](#_bookmark53) [C1CommandControl Command](#_bookmark54) [C1CommandMdiList Command](#_bookmark55)

###### C1CommandHolder Component

[C1CommandHolder](#_bookmark350) 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](#_bookmark243) 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

###### C1CommandMenu Command

The **C1CommandMenu** component is a command (derives from the [C1Command](#_bookmark243) base class) that is a menu. In addition to other [C1Command](#_bookmark243) properties, it contains a collection of command links which are the menu items of this menu. [C1CommandMenu](#_bookmark500) can be included in another menu as a sub-menu.

When a new [C1CommandMenu](#_bookmark500) 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.](#_bookmark114)

###### C1ContextMenu Control

The **C1ContextMenu** component is a menu (it derives from the [C1CommandMenu](#_bookmark500) base class) that can be attached to an arbitrary control as a context menu. To facilitate this, the [C1CommandHolder](#_bookmark350) (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](#_bookmark416) to the same **C1ContextMenu** in both places.

For more information about using the **C1ContextMenu**, please see [Context Menu Tasks.](#_bookmark150)

###### C1CommandControl Command

[C1CommandControl](#_bookmark306) is a command which can be associated with an arbitrary control. This functionality is provided by the class **C1CommandControl**, derived from [C1Command.](#_bookmark243) Controls can be dragged from the Visual Studio Toolbox and dropped onto a [C1MainMenu](#_bookmark773) or [C1ToolBar.](#_bookmark1170) 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.



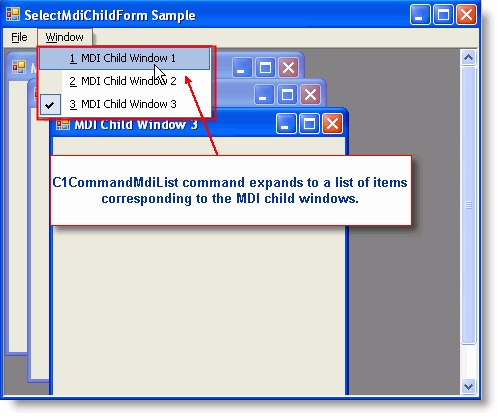
**Note:** A small control can be added inside it, but it does not handle large controls such as containers.

For more information about using [C1CommandControl,](#_bookmark306) see [Adding an Arbitrary Control to the](#_bookmark137) [Toolbar.](#_bookmark137)

###### C1CommandMdiList Command

[C1CommandMdiList](#_bookmark483) 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](#_bookmark483) displays a list of items corresponding to the MDI child windows.



You can restrict the amount of items the [C1CommandMdiList](#_bookmark483) command displays in its Menu's list by setting the [MaxItems](#_bookmark496) 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](#_bookmark495) 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.](#_bookmark126)

#### 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](#_bookmark773) control which is the main menu and the toolbars have a [C1ToolBar](#_bookmark1170) control which represents the toolbar.

The following section introduces the [C1MainMenu](#_bookmark773) and [C1ToolBar](#_bookmark1170) controls and provides further information about their appearance and behavior properties.

**See Also**

[C1MainMenu Control](#_bookmark57) [C1ToolBar Control](#_bookmark58)

###### C1MainMenu Control

[C1MainMenu](#_bookmark773) 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](#_bookmark350) will automatically appear in the component tray. The [C1CommandHolder](#_bookmark350) stores all of the menu's commands as a single collection. Only one [C1MainMenu](#_bookmark773) 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](#_bookmark773) component or drag and drop it onto the form.

**To add the C1MainMenu control programmatically:**

To write code in Visual Basic

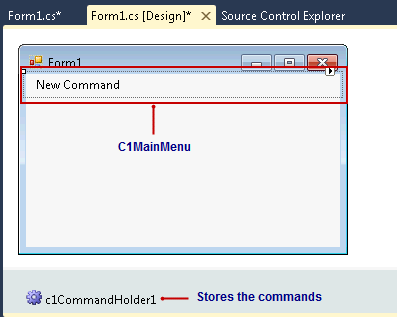
|  |  |
| --- | --- |
| Visual Basic | Copy Code |
| Imports C1.Win.C1Command  Dim ch As C1CommandHolder(Me) Dim mm As New C1MainMenu Me.Controls.Add(mm) | |

To write code in C#

C# Copy Code

using C1.Win.C1Command C1CommandHolder.CreateCommandHolder(this); C1MainMenu mm = new C1MainMenu(); this.Controls.Add(mm);

The following screen shot depicts a [C1MainMenu](#_bookmark773) control once it’s been added to the form:



The [C1MainMenu](#_bookmark773) 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](#_bookmark29) [Designer.](#_bookmark29)

For more information that shows how to use the **C1MainMenu** control for specific tasks, see [Menu](#_bookmark114) [Tasks.](#_bookmark114)

###### C1ToolBar Control

The [C1ToolBar](#_bookmark1170) control is used on forms as a toolbar. When you place this object on your form, like the [C1MainMenu,](#_bookmark773) a [C1CommandHolder](#_bookmark350) will automatically appear in the component tray. The [C1CommandHolder](#_bookmark350) 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](#_bookmark1170) represent buttons on the toolbar.

Once the component [C1ToolBar](#_bookmark1170) is added to the form, the **Link to Command** designer allows you to set up the toolbar system. [C1ToolBar](#_bookmark1170) and [C1MainMenu](#_bookmark773) both use the same **Link to Command** designer. For more information about the interface for the **Link to Command** designer, see [Link to](#_bookmark29) [Command Designer.](#_bookmark29)

The [C1ToolBar](#_bookmark1170) 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.](#_bookmark136)

#### 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](#_bookmark773) control.Menu Visual Styles

**See Also**

[Menus Visual Styles](#_bookmark60)

[Look and Feel of Menu Items](#_bookmark61) [Special Side Caption Styles in Menus](#_bookmark62) [Mouse-Over Styles in Menu Items](#_bookmark63) [Merging Menus](#_bookmark64)

[Layout and Text Wrapping in Menus](#_bookmark65) [ToolTips in Menus](#_bookmark66)

###### Menus Visual Styles

The [C1MainMenu](#_bookmark773) and [C1ContextMenu](#_bookmark576) 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](#_bookmark773) control. The [C1ContextMenu](#_bookmark576) control’s visual styles are identical to the [C1MainMenu](#_bookmark773) control’s, only the [C1ContextMenu](#_bookmark576) 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** |  |
| **VisualStyle.Office2010Blue** |  |

|  |  |
| --- | --- |
| **VisualStyle.Office2010Black** |  |
| **VisualStyle.Office2010Silver** |  |

###### Look and Feel of Menu Items

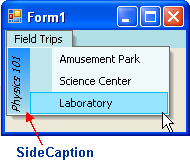
**C1Command** supports various settings for font style, border style, back color, and mouse-over styles for the menu items.

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](#_bookmark446) property overrides the **C1Command**[.Text](#_bookmark274) property.

###### Special Side Caption Styles in Menus

C1Command has a special [SideCaption](#_bookmark528) property for its [C1CommandMenu](#_bookmark500) command type. With this property, you can display a side caption along the sub menu items for a particular [C1CommandMenu.](#_bookmark500) 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](#_bookmark124) [Command Menu.](#_bookmark124)

###### 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](#_bookmark773) component has two special properties for applying mouse-over techniques.

The [BackHiColor](#_bookmark786) property gets the back color of the menu item when you hover your mouse over it and the [ForeHiColor](#_bookmark793) 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](#_bookmark132) [Menus.](#_bookmark132)

###### Merging Menus

In some cases, when you need to merge a MDI child window with a MDI parent menu you can enable the [CanMerge](#_bookmark788) property. You can also specify the type of behavior for the merging menu with its [MergeType](#_bookmark440) property. You can determine whether to add, replace, remove, or merge menu items with the [MergeType](#_bookmark440) 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](#_bookmark439) 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.](#_bookmark131)

###### Layout and Text Wrapping in Menus

[C1MainMenu](#_bookmark773) has an automatic layout. The menu items are automatically sized.

C1MainMenu's [Wrap](#_bookmark796) 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](#_bookmark773) provides a [ShowToolTips](#_bookmark794) property that displays the value of the [Text](#_bookmark274) 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](#_bookmark273) to **False**, and then setting custom text for the [ToolTipText](#_bookmark275) property.

For more information how to use the ToolTips, see [Displaying ToolTips for Menus and Toolbars.](#_bookmark129)

#### Toolbars Appearance and Behavior

[C1ToolBar](#_bookmark1170) provides a number of useful properties to control the behavior and appearance of the toolbars and toolbar buttons.

[C1ToolBar](#_bookmark1170) 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](#_bookmark1170) control.

**See Also**

[Toolbar Visual Styles](#_bookmark68) [Look and Feel of Toolbars](#_bookmark69)

[Special Border Styles in Toolbars](#_bookmark70) [Mouse-Over Styles in Toolbar Buttons](#_bookmark71) [Docking and Floating Toolbars](#_bookmark72) [Embedded Controls in Toolbars](#_bookmark73)

[Run-Time Customization for Toolbars](#_bookmark74) [Wrapping Toolbar Buttons and Text](#_bookmark75) [ToolTips in Toolbars](#_bookmark76)

[Toolbar and Button Layout Behavior](#_bookmark77)

###### Toolbar Visual Styles

The [C1ToolBar](#_bookmark1170) 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](#_bookmark1209) property.

The following table illustrates each of the [C1ToolBar](#_bookmark1170) 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](#_bookmark446) property overrides the **C1Command**. [Text](#_bookmark274) property.

For more information about using the general appearance properties, see [Modifying the](#_bookmark144) [Appearance of the Toolbar.](#_bookmark144)

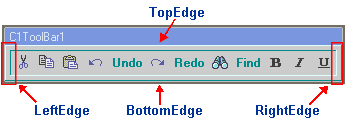
###### Special Border Styles in Toolbars

C1ToolBar has a special class, [C1Border,](#_bookmark214) that allows you to add various border styles to the toolbars. C1Border contains the following members:

|  |  |
| --- | --- |
| Name | Description |
| [BottomEdge](#_bookmark226) | Determines whether the border has a bottom edge. |
| [DarkColor](#_bookmark227) | Gets or sets the color of the group. In the Flat setting for the [Style](#_bookmark231) this color applies to the top, bottom, left, and right edges of the toolbar. |
| [LeftEdge](#_bookmark228) | Determines whether the border has a left edge. |
| [LightColor](#_bookmark229) | Gets or sets the color of the border. This color is not used in the Flat setting for the C1Border |
| [RightEdge](#_bookmark230) | Determines whether the border has a right edge. |
| [Style](#_bookmark231) | Gets or sets the border style. |
| [TopEdge](#_bookmark232) | Determines whether the border has a top edge. |
| [Width](#_bookmark233) | 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](#_bookmark231) 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](#_bookmark233) property is set to 5 pixels, [DarkColor](#_bookmark227) property is set to **DarkTurquoise**, [LeftEdge](#_bookmark228) property is set to PaleTurquoise, and

the [LeftEdge,](#_bookmark228) [RightEdge,](#_bookmark230) [BottomEdge,](#_bookmark226) and [TopEdge](#_bookmark232) are all set to **True**.

|  |  |
| --- | --- |
| Property Setting | Image |
| **Style.None** |  |
| **Style.Flat** |  |
| **Style.Groove** |  |
| **Style.Ridge** |  |
| **Style.Inset** |  |
| **Style.Outset** |  |

The following table illustrates the effect of the [LeftEdge,](#_bookmark228) [RightEdge,](#_bookmark230) [BottomEdge,](#_bookmark226) and [TopEdge](#_bookmark232)

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.](#_bookmark144)

###### 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](#_bookmark1170) component has two special properties for applying mouse-over techniques.

The [BackHiColor](#_bookmark1183) property gets the back color of the toolbar button when you hover your mouse over it and the [ForeHiColor](#_bookmark1203) gets the fore color of the toolbar button when you hover your mouse over it.

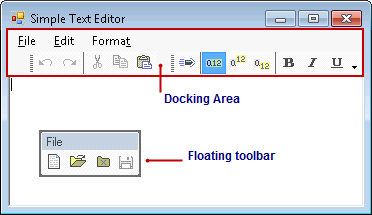
For more information about using these properties, see [Modifying the Appearance of the Toolbar.](#_bookmark144)

###### Docking and Floating Toolbars

Toolbars can be docked to the top, left, right or bottom on the container that the

[C1CommandDock](#_bookmark323) has been assigned to.

Each [C1ToolBar](#_bookmark1170) 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](#_bookmark1170) programmatically and would like to use the [C1CommandDock](#_bookmark323) 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 | Copy Code |
| Me.C1CommandDock = New C1.Win.C1Command.C1CommandDock() Me.C1CommandDock.Controls.Add(Me.C1ToolBar1) Me.Controls.Add(Me.C1CommandDock) | |

To write code in C#

|  |  |
| --- | --- |
| C# | Copy Code |
| this.c1CommandDock = new C1.Win.C1Command.C1CommandDock(); this.c1CommandDock.Controls.Add(this.c1ToolBar1); this.Controls.Add(this.c1CommandDock); | |

###### Embedded Controls in Toolbars

The [C1CommandControl](#_bookmark306) lets you embed arbitrary controls to the toolbar.

Arbitrary controls such as a textbox can be embedded in a [C1ToolBar](#_bookmark1170) through the use of the

[C1CommandControl.](#_bookmark306)

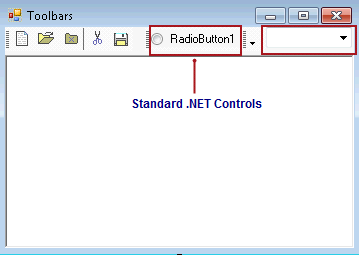
This can simply be done by dragging an arbitrary control on to the toolbar, adding a [C1CommandControl](#_bookmark306) command type through the designer, or adding a [C1CommandControl](#_bookmark306) command type programmatically.

For more information about how to embed an arbitrary control into the [C1ToolBar](#_bookmark1170) object, see

[Adding an Arbitrary Control to the Toolbar.](#_bookmark137)

When an arbitrary control is dragged to the toolbar it automatically creates a new command type called [C1CommandControl.](#_bookmark306) The [C1CommandControl](#_bookmark306) includes a [Control](#_bookmark317) 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.](#_bookmark1170)



###### Run-Time Customization for Toolbars

**C1ToolBars** can be customizable at run time by setting the [CustomizeButton](#_bookmark1199) property to **True** at design time.



**Note:** The toolbar needs to be placed inside a **C1CommandDock** before you set its

[CustomizeButton](#_bookmark1199) 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).](http://msdn2.microsoft.com/en-us/library/f432txa3.aspx)

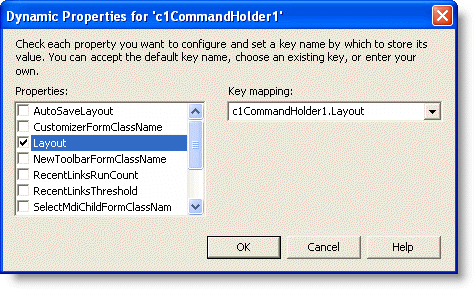
To Save the Layout in the Application's .Config File:

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.



1. 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

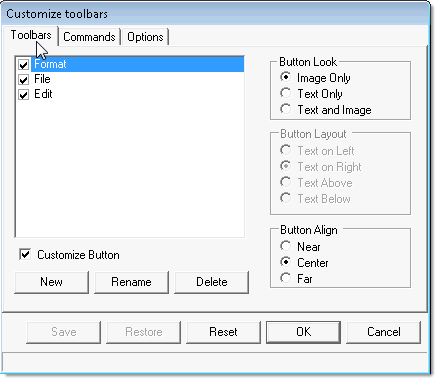
run the project in Visual Studio, change the toolbars layout, close it and then run it again, you won't see the last layout restored. This is not a bug, everything works fine when the application is not run from Visual Studio.

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](#_bookmark388) property in your code instead.



**Toolbars**

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



By default, the [ButtonLook](#_bookmark428) and the [CustomizeButton](#_bookmark1199) properties are disabled.

The Button Layout properties are enabled when the [ButtonLook](#_bookmark428) property is set to **Text and Image**(the **Text and Image** radio button is selected). This is because the [ButtonLayoutHorz](#_bookmark1188) [property](#_bookmark1188) determines how the text is placed by the image (above, below, to the left, or to the right of the image).

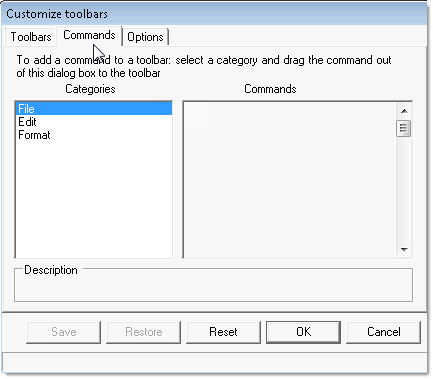
The [CustomizeButton](#_bookmark1199) is enabled when a new [C1ToolBar](#_bookmark1170) is added to the dialog box.

**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.



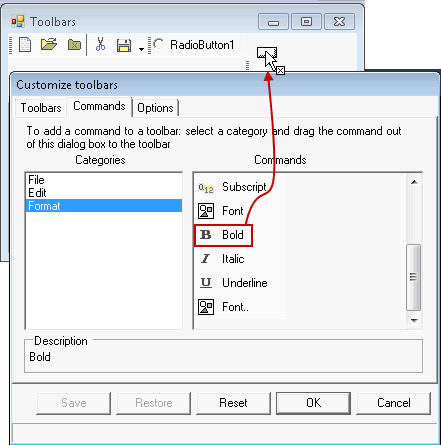
**Note:** The Categories list box appears empty if the [Category](#_bookmark257) property is not set for the commands.



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

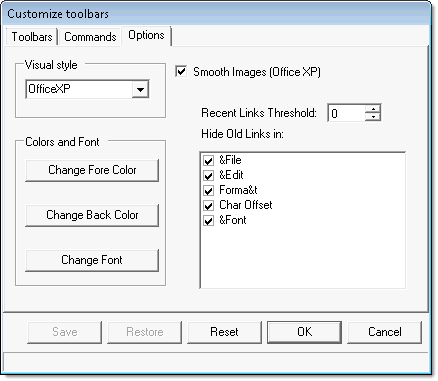
* + Selecting a category from the **Categories** list box.
  + Selecting a command from the **Commands** list box and then dragging it to the desired toolbar.

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



**Options**

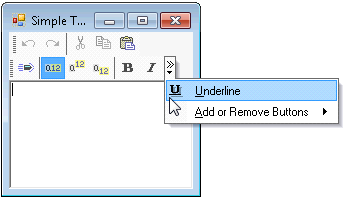
The **Options** tab contains options for modifying **C1ToolBar**’s general appearance properties such as its look and feel and its colors and font.



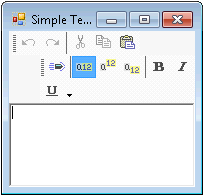
###### Wrapping Toolbar Buttons and Text

[C1ToolBar](#_bookmark1170) provides wrapping ability for toolbar buttons as well as wrapping text in the toolbar buttons. The [Wrap](#_bookmark1210) 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](#_bookmark1210) property set to **False**.



The following image shows how the toolbar buttons appear when its [Wrap](#_bookmark1210) property set to **True**.

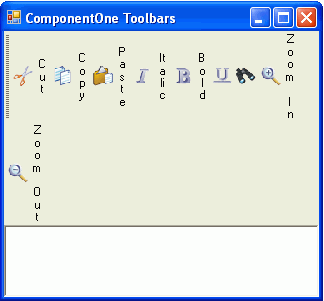


The following image shows how the toolbar buttons appear when its [Wrap](#_bookmark1210) is set to **True** and its

[WrapText](#_bookmark1211) properties is set to **False**.

The following image shows how the toolbar buttons appear when their [Wrap](#_bookmark1210) and [WrapText](#_bookmark1211)

properties are set to **True**.



###### ToolTips in Toolbars

A ToolTip is used to display text when the mouse hovers over the control. [C1ToolBar](#_bookmark1170) provides

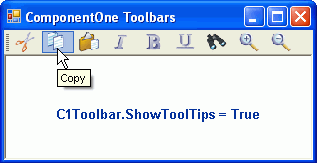
a [ShowToolTips](#_bookmark1207) property that displays the value of the [Text](#_bookmark274) property as a ToolTip for each toolbar button. This property is enabled by default.

**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.

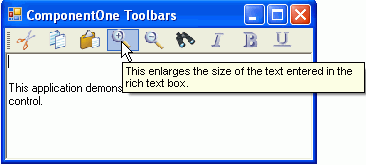
If you would like to enter custom text for the ToolTip of each toolbar button you can through the [ToolTipText](#_bookmark275) property.



The following image depicts a toolbar that has its [ShowToolTips](#_bookmark1207) property set to **True**.



The following image shows a toolbar that has its [ShowToolTips](#_bookmark1207) property set to **True** and custom text entered for fourth [C1CommandLink](#_bookmark416)'s [ToolTipText](#_bookmark275) property.



For more information about using ToolTips, see [Displaying ToolTips for Menus and Toolbars.](#_bookmark129)

###### Toolbar and Button Layout Behavior

[C1ToolBar](#_bookmark1170)s layout is very flexible. They can be horizontal or vertical as well as docked to specific areas of the form. The toolbar's [Movable](#_bookmark1206) 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](#_bookmark1204) to **False**.



**Note:** When you set the [ToolBarStyle](#_bookmark1208) property to DropDownMenu, the menu behaves like a drop-down so the toolbar becomes stationary.

In addition to toolbar orientation, [C1ToolBar](#_bookmark1170) also provides button alignment for vertical toolbars. You can align the image or text near, center, or far from the button through the [ButtonAlign](#_bookmark1186) property.

The following table shows the values for the [ButtonAlign](#_bookmark1186) property:

|  |  |
| --- | --- |
| Property Setting | Image |
| **ButtonAlign.Near** |  |
| **ButtonAlign.Center** |  |
| **ButtonAlign.Far** |  |

You can determine the relative position of text and images for toolbar buttons in horizontal and

vertical toolbars using the [ButtonLayoutHorz](#_bookmark1188) and [ButtonLayoutVert](#_bookmark1189) properties.

The [ButtonLayoutHorz](#_bookmark1188) property gets the layout of the buttons when the toolbar is horizontal. This is the default orientation of the toolbar. The [ButtonLayoutVert](#_bookmark1189) property gets the layout of the buttons when the toolbar is vertical. Setting the [Horizontal](#_bookmark1204) property to **False** gets the vertical orientation for the toolbar.

**Note:** The default value for the [ButtonLayoutHorz](#_bookmark1188) property is TextOnRight.

[C1ToolBar](#_bookmark1170) provides several options for customizing the toolbar buttons for vertical and horizontal toolbars.



|  |  |
| --- | --- |
| Property Setting | Image |
| **ButtonLayoutHorz.TextOnRight (default)** |  |
| **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](#_bookmark1194) property to display text, images, or both for the horizontal toolbar and the [ButtonLookVert](#_bookmark1195) property to display text, images, or both for the vertical toolbar.



**Note:** The Text, Image, and TextAndImage values for the [ButtonLook](#_bookmark428) property overrides the

values for the [ButtonLookHorz](#_bookmark1194) and [ButtonLookVert](#_bookmark1195) properties. The [ButtonLook](#_bookmark428) property should be set to default if you plan on setting values for the [ButtonLookHorz](#_bookmark1188) or [ButtonLookVert](#_bookmark1195) property.

The following table shows the values for the [ButtonLayoutVert](#_bookmark1189) property:

|  |  |
| --- | --- |
| Property Setting | Image |
| **ButtonLayoutVert.TextOnRight** |  |
| **ButtonLayoutVert.TextOnLeft** |  |
| **ButtonLayoutVert.TextAbove** |  |

**ButtonLayoutVert.TextBelow (default)**

