

VCSKicks CommandLink

<http://www.vcskicks.com/components/command-link.php>

How to Add VCSKicks CommandLink to Projects.....	2
How to Add VCSKicks CommandLink to the ToolBox	2
Supported Properties.....	3
Color.....	3
Text	4
Image	5
Other	5
Custom Events	6
Overridable Methods.....	7
Custom Events	7
Helper Functions.....	7
How Events are Triggered.....	8
Drawing Event Sequence	9
Frequently Asked Questions (F.A.Q.).....	9

How to Add VCSKicks CommandLink to Projects

On any given project, follow these steps:

1. Go to the Project menu
2. Select "Add Reference..."
3. In the Add Reference dialog, under the .NET tab, scroll down to find VCSKicksCommandLink. Select it and choose OK

How to Add VCSKicks CommandLink to the ToolBox

The following steps will add VCSKicksCommandLink to the Visual Studio ToolBox:

1. Open the ToolBox (View > ToolBox)
2. Right-Click and select "Choose Items..."
3. In the dialog, under the tab ".NET Framework Components" find "CommandLink" in the list of components.
4. If CommandLink **does not** appear on the list, click the button "Browse..." and find VCSKicksCommandLink.dll in the installation folder (Program Files\Visual C# Kicks\VCSKicksCommandLink\Binary by default)
5. Make sure the checkbox in the list is **checked** and click OK

Supported Properties

Color

AutoNormalBackgroundColor

Gets or sets whether the background color of the control in a *normal* state will be automatically set according to the system or manually set by the NormalBackgroundColor property.

AutoPushedBackgroundColor

Gets or sets whether the background color of the control in a *pushed* state will be automatically set according to the system or manually set by the PushedBackgroundColor property.

AutoPushedOutlineColor

Gets or sets whether the outline color of the control in a *pushed* state will be automatically set according to the system or manually set by the PushedOutlineColor property.

AutoPushedShadowColor

Gets or sets whether the shadow (inner outline) color of the control in a *pushed* state will be automatically set according to the system or manually set by the PushedShadowColor property.

AutoHoverDarkOutlineColor

Gets or sets whether the darker outline color of the control in a *hover* state will be automatically set according to the system or manually set by the HoverDarkOutlineColor property.

AutoHoverLightOutlineColor

Gets or sets whether the lighter outline color of the control in a *hover* state will be automatically set according to the system or manually set by the HoverLightOutlineColor property.

AutoHoverShineColor

Gets or sets whether the shine color of the control in a *hover* state will be automatically set according to the system or manually set by the HoverShineColor property.

AutoHighlightColor

Gets or sets whether the highlight color of the control when it is focused will be automatically set according to the system or manually set by the HighlightColor property.

NormalBackgroundColor

Gets or sets the background color of the control in a *normal* state that appears if the AutoNormalBackgroundColor property is set to false.

PushedBackgroundColor

Gets or sets the background color of the control in a *pushed* state that appears if the `AutoPushedBackgroundColor` property is set to false.

PushedOutlineColor

Gets or sets the outline color of the control in a *pushed* state that appears if the `AutoPushedOutlineColor` property is set to false.

PushedShadowColor

Gets or sets the shadow (inside outline) color of the control in a *pushed* state that appears if the `AutoPushedShadowColor` property is set to false.

HoverDarkOutlineColor

Gets or sets the darker outline color of the control in a *hover* state that appears if the `AutoHoverDarkOutlineColor` property is set to false.

HoverLightOutlineColor

Gets or sets the lighter outline color of the control in a *hover* state that appears if the `AutoHoverLightOutlineColor` property is set to false.

HoverShineColor

Gets or sets the shine color of the control in a *hover* state that appears if the `AutoHoverShineColor` property is set to false.

HighlightColor

Gets or sets the highlight color of the control when it is focused if the `AutoHighlightColor` property is set to false.

Text**WordWrapHeader**

Gets or sets whether text in the header will be automatically word wrapped when it goes outside the bounds of the control.

WordWrapDescription

Gets or sets whether text in the description will be automatically word wrapped when it goes outside the bounds of the control.

HeaderText

Gets or sets the bigger text of the control.

Description Text

Gets or sets the smaller text of the control.

Font

The font used to display the header text in the control. The description text is set to the same font, 3 pixels smaller.

DescriptionTextColor

Gets or sets the color used to display the description text in the control.

TextWordWrapMode

Gets or sets how text is word-wrapped. Text can be split up by words or by characters.

Text

The Text property is overridden to redirect to the HeaderText property.

CenterText

Gets or sets whether to center the text horizontally in the control.

TextImagePadding

Gets or sets the amount of padding in pixels between the control's Image and Texts.

Image**ImageScalingSize**

Gets or sets the target display dimensions of the image.

ImageAlign

Gets or sets the vertical alignment of the image.

ImageLeft

Adjusts the distance in pixels of the image from the side of the control.

HasImage [Not Visible in Designer]

Gets whether the control has an image associated with it.

Other**AutoEllipsis**

Enables the automatic handling of text that extends beyond the width of the control.

UseMnemonic

If true, the first character preceded by an ampersand (&) will be used as the control's mnemonic key.
Applies to HeaderText and DescriptionText

RightToLeft

Indicates whether the component should draw right-to-left for RTL languages.

Custom Events**DrawDesignTime**

Triggered after elements are drawn at design-time.

DrawBackground

Triggered after elements are drawn in the background of the control.

DrawHighlight

Triggered after elements are drawn for the highlight of the control.

DrawHoverState

Triggered after elements are drawn when the control is in a *hover* state.

DrawPushedState

Triggered after elements are drawn when the control is in a *pushed* state.

DrawNormalState

Triggered after elements are drawn when the control is in a *normal* state.

DrawForeground

Triggered after elements are drawn for the foreground of the control.

DrawImage

Triggered after the image is drawn on the control.

DrawText

Triggered after the texts are drawn on the control.

Overridable Methods

Custom Events

On[Custom Event]

All custom events can be overridden. For example, DrawText can be overridden with OnDrawText.

Helper Functions

GetNormalBackgroundColor

Calculates the background color of the control in a *normal* state depending on the system colors.

GetPushedBackgroundColor

Calculates the background color of the control in a *pushed* state depending on the system colors.

GetPushedOutlineColor

Calculates the outline color of the control in a *pushed* state depending on the system colors.

GetPushedShadowColor

Calculates the shadow (inner outline) color of the control in a *pushed* state depending on the system colors.

GetHoverDarkOutlineColor

Calculates the darker outline color of the control in a *hover* state depending on the system colors.

GetHoverLightOutlineColor

Calculates the lighter outline color of the control in a *hover* state depending on the system colors.

GetHoverShineColor

Calculates the shine color of the control in a *hover* state depending on the system colors.

GetHighlightOutlineColor

Calculates the highlight color of the control when it is focused depending on the system colors.

GetRoundedRectangle

Returns a rounded rectangle GraphicsPath.

GetGrayscale

Returns the grayscale representation of an image.

WordWrap

Returns a string that fits within the target width. Excess words or characters are moved to a new line.

ClipString

Returns a string with characters outside the target width removed.

ReplaceLastCharacters

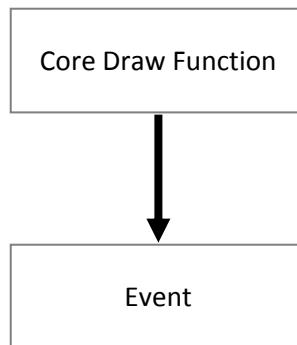
Returns a string with the last given number of characters replaced with a new character.

GetFormattedHeader

Formats the header text based on control properties.

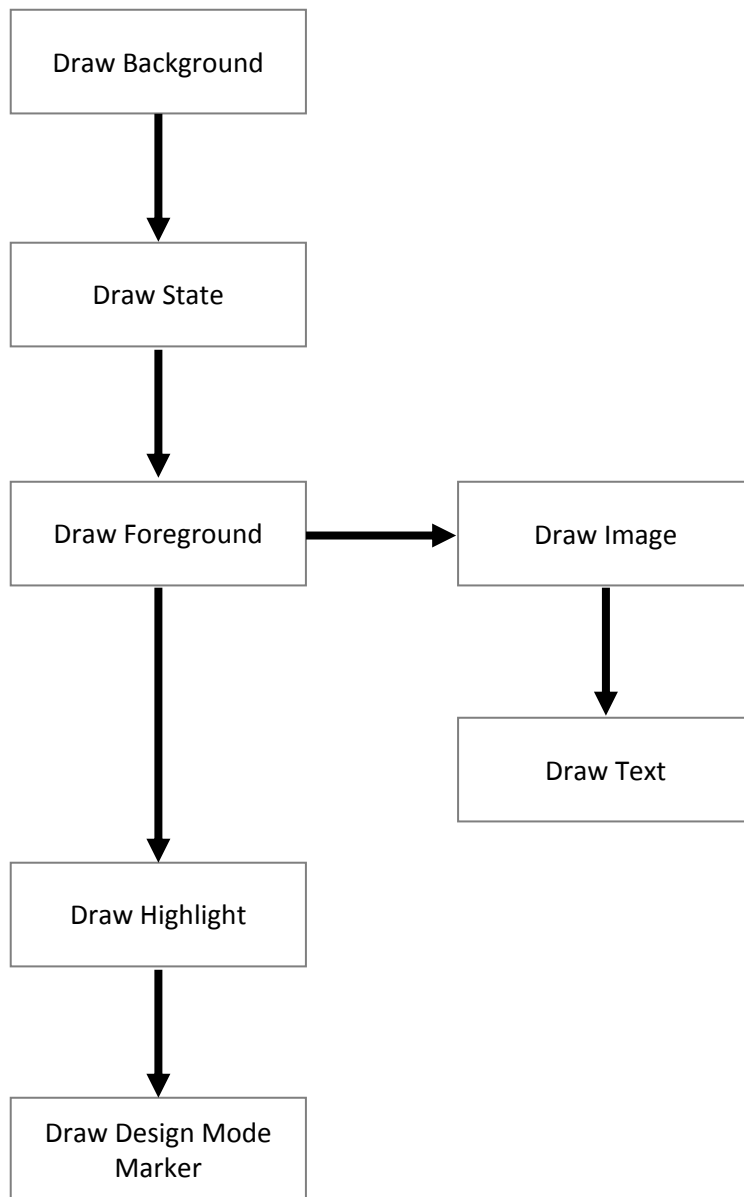
GetFormattedDescription

Formats the description text based on control properties.

How Events are Triggered

Note: If a custom drawing event is overridden, events will no longer be called unless done explicitly.

Drawing Event Sequence



Frequently Asked Questions (F.A.Q.)

Why is there a dotted blue outline on the control?

The dotted blue outline of the control is drawn only at design-time and is intended to make it easier for developers to layout the control.

What is the difference between DrawBackground and DrawNormalState?

DrawBackground draws the solid back color of the control and then draws a rounded rectangle on top of that. DrawNormalState does nothing by default. DrawBackground is always called, while DrawNormalState is only called when the control is a normal state (i.e. not in a pushed or hover state).

What is the difference between BackColor and NormalBackgroundColor?

The BackColor property refers to the solid color drawn at the very back of the control. NormalBackgroundColor is used for drawing the rounded rectangle of the control.

Does the control support a transparent BackColor?

No. In order to keep optimal performance, a transparent BackColor is not supported.

Can the control be inherited to create variations of the control?

Yes. Most elements of the control are split into functions that can be overridden when inheriting the control. This makes it possible to create subtle variants as well as drastically different controls.

What class does the control inherit?

VCSKicksCommandLink inherits System.Windows.Forms.Button.