VSVBX 4.0

The VideoSoft Custom Control Library

Contents

Introduction

Welcome to VSVBX 4.0, the VideoSoft Custom Control Library. If you are upgrading from a previous version, make sure to look in the New Features in VSVBX 4.0 section.

The VideoSoft Controls were designed to save you from writing tedious, repetitive, error-prone code. The controls are innovative and efficient. They are distributed as a single VBX to make installation easier.

Our distribution policy is almost as innovative as the controls. We want every Visual Basic programmer to get a copy of VSVBX and try it for as long as they want. Those who like the product and find it useful (almost everybody, we hope) can buy a license for a reasonable price. The only restriction is that unlicensed copies of VSVBX display a VideoSoft banner whenever they are loaded, to remind developers to license the product.

We hope you'll like VSVBX. If you have suggestions and ideas for new features or new controls, call us or write.

VideoSoft
2625 Alcatraz Avenue, Suite 271
Berkeley, CA 94705
(510) 547-7295 (phone)
(510) 547-1084 (fax)

Control Summary

Icon	Name	Object	Description
	Elastic	VSElastic	Smart containers that resize themselves and their child controls, automatically create labels and 3-D frames for its child controls, and can also be used as progress indicators and labels.
	IndexTab	VSIndexTab	Allow you to group controls by subject, using the familiar notebook metaphor that has become a Windows standard.
	Awk	VSAwk	Parsing engine named and patterned after the popular Unix utility, plus a powerful expression evaluator.

Installation

To install VSVBX, just copy the following files to your WINDOWS\SYSTEM directory:

VSVBX.VBX — This file contains the controls. To use VSVBX from Visual Basic, you must include this file in your project.

VSVBX.LIC — This is the **run-time** license file. If VSVBX cannot find this file when called from a compiled VB application, it displays a VideoSoft banner and waits for the user to click Ok.

VSVBX.DEV — This is the **developers** license file. If VSVBX cannot find this file when loaded into Visual Basic, it displays a VideoSoft banner and waits for the user to click Ok.

VSVBX.HLP — This file contains the VSVBX on-line help.

Note: The VBX, LIC, DEV, and HLP files should always have the same name. Some people like to rename VBX files when they distribute an application. If you rename the VSVBX.VBX file, you must also rename the license and help files (e. g. FOO.VBX, FOO.LIC, FOO.DEV, FOO.HLP).

If you would rather install VSVBX in a different directory, thats fine. As long as the help and license files are either in the same directory as the VBX, in the WINDOWS\SYSTEM directory, or in the WINDOWS directory, VSVBX will find them for you.

Distribution

VSVBX is royalty-free. You may include copies of the VBX and LIC files with as many copies of as many applications you ship.

You cannot distribute the developers license file VSVBX.DEV.

Site Licenses

If you work with other developers, you may be interested in VideoSofts *Power Of 2* site licenses. It works this way: when you purchase *n* licenses, you are allowed to install the software on *n2* machines. A development team with 25 people, for example, would need to buy only 5 licenses.

Product Support

Product support for VSVBX is available to licensed users through the following channels:

CompuServe: CIS 71552,3052

Mail: VideoSoft
2625 Alcatraz Avenue, Suite 271
Berkeley, California 94705

Phone: (510) 547-7295
Fax: (510) 547-1084

Before calling for technical support, please make sure you know what version of VSVBX you are using. The version number appears in the About box that pops up when you double-click the About property in any of the VSVBX controls.

New Features in VSVBX 4.0

This section summarizes the new features in VSVBX 4.0. For details on each new feature, check the main body of the documentation. Most of these enhancements were added in response to user requests, and we are grateful for their input.

VSVBX 4.0 is a vast improvement over previous versions. It is easier to use, faster, the controls look better, and it has incredible new functionality.

We have also improved the documentation to make it more accurate, complete, and accessible to new users. Check out the TroubleShooting and QuickStart sections on the IndexTab, Elastic, and Awk.

New Features: Elastic, IndexTab, Awk.

Elastic 4.0

The new Elastic features more options for the frame bevels, a new frame style that allows you to show the caption over the frame, and more flexibility for the unique tag labels.

New Properties

Style allows you to create Elastics with captions drawn over the outer frame, on the top or on the bottom of the control. This new property makes the Elastic look similar to the standard frame control, but retaining its resizing and 3D capabilities.

TagPosition allows the tag labels to be placed either to the left or above each tagged control. In previous versions, the tag labels always appeared to the left of the tagged controls. Label sizes are still determined by the **TagWidth** property.

ShadowColor allows you to define the color for the shadow created when the **BevelInner** property is set to 7 - Shadow.

New Settings

The **BevelInner** and **BevelOuter** properties have the following new settings: 5 Fillet, 6 Groove, and 7 Shadow 3D effects.

IndexTab 4.0

The new IndexTab features the new standard look for tabs, like those found in Microsofts Word and Excel. Access keys may now be used regardless of tab position, there is improved mouse tracking, dog-ears for easier scrolling, pictures and custom colors for each tab, automatic bolding of the current tab caption, multiple tab rows and more.

The new IndexTab offers design time tab switching, making it much easier to design attractive and functional forms. When **AutoSwitch** is True, double-click on any tab with the **RIGHT** mouse button to bring up the appropriate child container, just like at runtime. Then you can easily lay out each container. To see all containers, double-click on the blank space next to the last tab or on the current tab itself. All containers will appear side-by-side, arranged according to their tab sequence. To change the tab sequence, just drag the controls around. To disable this feature, set **AutoSwitch** to False.

New properties

BoldCurrent causes the IndexTab to bold the caption of the current tab.

BorderWidth determines the width of the border around the containers inside the IndexTab. This border is in addition to the bevel automatically provided when one of the 3D styles is used.

DogEars determines whether the IndexTab should display dog-ears at the start and end of a tab row to indicate the user can scroll the tabs. If this property is set to True, the dog-ears appear automatically when the tabs spill off the page and you can use the mouse to scroll the tabs.

FrontTabForeColor lets you specify the color for the caption of the currently selected tab. You can use this property to make the current tab stand out.

MouseOver returns the number of the tab being pointed at by the mouse. You can use this property and the standard MouseMove event to provide dynamic tab help.

MultiRow determines whether the IndexTab should display multiple tab rows instead of letting them scroll off the screen. When this property is set to True, tab scrolling is automatically disabled. This property can only be used when the **TabsPerPage** property is set to a number greater than zero.

TabPicture, **TabColor**, and **TabEnabled** are property arrays that allow you to attach a picture (bitmap or icon) to each tab, to control the color of individual tabs, and to enable and disable specific tabs so they cannot be selected with the mouse or keyboard.

New Settings

The **Style** property has two new settings: 6 Straight and 7 Straight 3D.

Awk 4.0

The new Awk has a couple of new properties that make it even easier to parse text strings, plus a brand new, high-powered expression evaluator that can be used in everything from letting your users type in formulas instead of values to sophisticated scientific applications.

New Properties

FilterQuotes allows you to automatically eliminate quotes from Awk fields.

Val gives the Awk a powerful expression evaluator with many built-in functions and support for user-defined variables. Use this feature to allow your users to enter formulas directly into your application instead of forcing them to use a calculator. Or use the support for user-defined variables to build complex scientific applications.

Elastic Reference

Description The VideoSoft Elastic control is a versatile smart container. It can save you hundreds of lines of VB code by allowing you to:

1. Automatically resize the Elastic to the left, right, top, bottom, or to fill its container, be it a form or another control.

2. Automatically resize the Elastics child controls, evenly or unevenly, vertically, horizontally, or proportionally.

3. Label child controls based on their Tag property, instead of using several Label controls.

4. Allow the user to resize controls inside the Elastic at run time, using the mouse (Splitter bars).

5. Create multi-line labels, 3-D gauges, or both at the same time in a single Elastic.

6. Give plain controls a 3-D look.

File Name VSVBX.VBX

Object Type VSElastic

Remarks The maximum number of children an Elastic can resize is 30. If you need more, use nested Elastics. Or you may be able to cut down the number of children you need by using the **TagWidth** property.

If you set an Elastics **BorderWidth** and **ChildSpacing** properties to 0 and fill it with other controls, it may become impossible to select it with the mouse. You can still select it either by cycling through the controls with the tab key or by choosing it from the drop-down list box on top of the Properties Window.

Changing certain properties in the Text and List controls forces them to be destroyed and recreated. If they are inside an Elastic with the **AutoSizeChildren** property set to uneven spacing, other controls will expand to fill the Elastic when the Text or List controls are destroyed; when they are recreated, there will be no room for them, so their size will be reset to zero.

Elastic Summary

Properties (default: Caption)

(About)	* AccessKey [3]	* Align
* AutoSizeChildren	BackColor	* BevelChildren [3]
* BevelInner	* BevelInnerWidth	* BevelOuter
* BevelOuterWidth	* BorderWidth	Caption
* CaptionPos	* ChildSpacing	DragIcon
DragMode	Enabled	* FloodColor
* FloodDirection	* FloodPercent	FontBold
FontItalic	FontName	FontSize
FontStrikethru	FontUnderline	ForeColor
Height	HelpContextID [3]	hWnd [3]
Index	Left	* MaxChildSize [3]
* MinChildSize [3]	MousePointer	Name
Parent	* ShadowColor [4]	* Splitter
* Style [4]	TabIndex	TabStop
Tag	* TagPosition [4]	* TagWidth [3]
Top	Visible	Width
* WordWrap [3]		

Events

Click	DblClick	DragDrop
DragOver	KeyDown	KeyPress
KeyUp	MouseDown	MouseMove
MouseUp	* RealignFrame	* ResizeChildren

Methods

Move	Refresh	SetFocus
Zorder		

AccessKey Property [3.0]

Description Determines whether the Elastic should show and process access keys. Access keys are created by preceding characters in the **Caption** with an ampersand (&).

Usage [*form.*]*VSElastic*.**AccessKey** [= {**True**|**False**}]

Remarks If this property is set to True, the user can use alt-key combinations to move the focus in a form. If it is set to False, ampersands are displayed as regular text in the Caption and alt-key combinations are ignored.

This property affects how the Caption is displayed and also the tag labels if the TagWidth property is set to a non-zero value.

Default Value False

Data Type Boolean

Align Property

Description Sets or returns the automatic alignment setting for the Elastic.

Usage [*form*.]*VSElastic*.**Align** [= *setting%*]

Remarks Valid options for this property are:

0 - No automatic alignment
1 - Top
2 - Bottom
3 - Left
4 - Right
5 - Fill Container

The Elastics **Align** property is similar to the standard Align property, except for two things:

> The Elastics Align has three additional settings: Left, Right, and Fill Container.

> The Elastics Align works even when the Elastic is inside another control. The standard VB Align property only works when the control is placed on the form.

Default Value 0 - No automatic alignment

Data Type Integer

AutoSizeChildren Property

Description Determines how the Elastic resizes the controls inside it.

Usage [*form*.]*VSElastic*.**AutoSizeChildren** [= *setting%*]

Remarks Valid options for this property are:

0 - No automatic child sizing
1 - Even Horizontally
2 - Uneven Horizontally
3 - Even Vertically
4 - Uneven Vertically
5 - Elastics Horizontally [3.0]
6 - Elastics Vertically [3.0]
7 - Proportional [3.0]

The *Even Horizontally and Even Vertically* options make all controls in the Elastic have the same width or height, respectively, and spread them evenly across the Elastic.

The *Uneven Horizontally* and *Uneven Vertically* options only stretch the front control inside the Elastic (highest Z-Order). The other controls are spread across the Elastic, but they are not resized.

The *Elastics Horizontally* and *Elastics Vertically* options only stretch other Elastics. These settings can be used to center controls and to form clusters of controls inside an Elastic.

The *Proportional* setting keeps the relative size and position of all child controls constant when the Elastic is resized. While this setting is active, all controls in the Elastic become locked; you can't move them because the Elastic will put them back where they were. To change the layout of the form, you first have to set the AutoSizeChildren property to *None.*

The maximum number of children an Elastic can resize is 30. If you need more, use nested Elastics.

Default Value 0 - None

Data Type Integer

BevelInner, BevelOuter Property

Description Sets or returns the 3-D appearance of the Elastic and of the controls inside it.

Usage [*form.*]*VSElastic*.**BevelInner** [= *setting%*]
[*form.*]*VSElastic*.**BevelOuter** [= *setting%*]

Remarks Valid options for this property are:

0 - None
1 - Raised
2 - Raised outlined
3 - Inset
4 - Inset outlined
5 - Fillet [4.0]
6 - Groove [4.0]
7 - Shadow [4.0] (available for **BevelInner** only)

Default Value

BevelOuter:	2 - Raised Outlined
BevelInner:	3 - Inset

Data Type Integer

BevelInnerWidth, BevelOuterWidth Property

Description Sets or returns the thickness of the bevels defined by the **BevelInner** and **BevelOuter** properties. Units are pixels.

Usage [*form*.]*VSElastic*.**BevelInnerWidth** [= *value%*]
[*form*.]*VSElastic*.**BevelOuterWidth** [= *value%*]

Remarks The thickness of the bevels is limited by the Elastics **BorderWidth.**

Default Value BevelOuterWidth: 2
BevelInnerWidth: 1

Data Type Integer

BevelChildren Property [3.0]

Description Determines whether the Elastic should draw 3-D frames around all its child controls or only around specified types of controls.

Usage [*form*.]*VSElastic*.**BevelChildren** [= *setting%*]

Remarks Valid settings for this property are:

0 - All
1 - No Graphical
2 - No Elastics
3 - No Graphical or Elastics

This property adds control over the look of forms that contain labels and edit controls. For example, to make labels look flat and edit controls look raised, set BevelChildren to 1 (labels are graphical controls).

Default Value 0 - All

Data Type Integer

BorderWidth Property

Description Sets or returns the width of the border around the child controls and the Elastic.

Usage [*form*.]*VSElastic*.**BorderWidth** [= *setting%*]

Remarks The border width is measured in pixels.

Default Value 6

Data Type Integer

CaptionPos Property

Description Sets or returns the position of the Elastics caption. Also sets the position of the tag labels, if they are enabled (see the TagWidth property).

Usage [*form*.]*VSElastic*.**CaptionPos** [= *setting%*]

Remarks Valid options for this property are:

0 - Left Top
1 - Left Center
2 - Left Bottom
3 - Center Top
4 - Center Center
5 - Center Bottom
6 - Right Top
7 - Right Center
8 - Right Bottom

Default Value 1 - Left Center

Data Type Integer

ChildSpacing Property

Description Sets or returns the spacing between the Elastics child controls and between controls and their tag labels.

Usage [*form*.]*VSElastic*.**ChildSpacing** [= *setting%*]

Remarks The child spacing is measured in pixels.

If the AutoSizeChildren property is set to zero, this property has no effect.

Default Value 6

Data Type Integer

FloodColor Property

Description Sets or returns the color used to fill the Elastic when it is used as a progress indicator.

Usage [*form*.]*VSElastic*.**FloodColor** [= *color&*]

Remarks If **FloodDirection** is set to zero, this property has no effect.

Default Value Red

Data Type Long

FloodDirection Property

Description Sets or returns the direction used to fill the Elastic when it is used as a progress indicator.

Usage [*form*.]*VSElastic*.**FloodDirection** [= *setting%*]

Remarks Valid settings for this property are:

0 - None
1 - Fill Right
2 - Fill Left
3 - Fill Up
4 - Fill Down

Default Value 0 - None

Data Type Integer

FloodPercent Property

Description Sets or returns the percentage of the Elastic that should be flooded when it is used as a progress indicator.

Usage [*form*.]*VSElastic*.**FloodPercent** [= *setting%*]

Remarks Setting this property to a value smaller than 0 or greater than 100 will not cause any errors.

If FloodDirection is set to zero, this property has no effect.

Default Value Zero

Date Type Integer

MaxChildSize Property [3.0]

Description Sets or returns the maximum size for the Elastics child controls. The size is measured in twips and may correspond to the width or height of the child control, depending on the setting of the AutoSizeChildren property.

Usage [*form*.]*VSElastic*.**MaxChildSize** [= *value&*]

Remarks The maximum size limit only applies to the child controls being resized by the Elastic. If AutoSizeChildren is set to Uneven Spacing, for example, only the top child is affected.

This property is useful to prevent controls inside an Elastic from spreading too much. Command buttons, for example, are harder to use if they are too far apart from each other.

Setting this property to 0 disables it.

Default Value Zero

Date Type Long

MinChildSize Property [3.0]

Description Sets or returns the minimum size for the Elastics child controls. The size is measured in twips and may correspond to the width or height of the child control, depending on the setting of the AutoSizeChildren property.

Usage [*form*.]*VSElastic*.**MinChildSize** [= *value&*]

Remarks The minimum size limit only applies to the child controls being resized by the Elastic. If AutoSizeChildren is set to uneven spacing, for example, only the top child is affected.

This property is useful to prevent controls inside an Elastic from getting too narrow. It may be better, for example, to show only a few command buttons with readable captions than to show all of them truncating the captions.

This property can be used in conjunction with the Splitter property to prevent users from making a child control too small.

Setting this property to 0 disables it.

Default Value Zero

Date Type Long

RealignFrame Event

Description Fired after the Elastic is resized.

Syntax **Sub** *CtlName*_**RealignFrame ()**

ResizeChildren Event

Description Fired after the Elastics children are resized.

Syntax **Sub** *CtlName*_**ResizeChildren ()**

ShadowColor Property [4.0]

Description Determines the color used to display the shadow of the child controls when the BevelInner property is set to 7 Shadow.

Usage [*form*.]*VSElastic*.**ShadowColor** [= *colorref&*]

Default Value Dark Gray

Date Type Color

Splitter Property

Description Determines whether the user is allowed to resize Elastic child windows by dragging the bar between two child windows.

Usage [*form*.]*VSElastic*.**Splitter** [= {**True|False**}]

Remarks If the Splitter property is set to True, the Elastic will monitor the mouse at run time. Whenever the mouse passes over the space between controls inside the Elastic, the pointer changes to a resize icon and the user is allowed to drag the border between the adjacent controls to a new location.

To use the **Splitter** property, the **ChildSpacing** property must be set to *Uneven Vertical* or *Uneven Horizontal*.

Default Value False

Data Type Boolean

Style Property [4.0]

Description Determines whether the Caption should be positioned in the Elastics client area or over its frame, similar to the standard frame control.

Usage [*form*.]*VSElastic*.**Style** [= *setting%*]

Remarks Valid settings for this property are:

0 - Classic
1 - Frame Top
2 - Frame Bottom

Default Value 0 - Classic

Date Type Integer

TagPosition Property [4.0]

Description Determines whether the tag labels should be displayed to the left or above the tagged controls. For details on using tag labels, see the TagWidth property.

Usage [*form*.]*VSElastic*.**TagPosition** [= *setting%*]

Remarks Valid settings for this property are:

0 - Left of Control
1 - Above Control

Default Value 0 - Left of Control

Date Type Integer

TagWidth Property [3.0]

Description This property allows you to use the Tag property of the Elastics children as labels, instead of using label controls. This facilitates form design and can also save lots of labels.

Usage [*form.*]*VSElastic*.**TagWidth** [= *setting&*]

Remarks Tag labels are always placed to the left of the control being labeled, and their height is the same as that of the control being labeled. The width of the labels is controlled by the TagWidth property, as explained below.

Positive TagWidth values are interpreted as the width of the tag labels, measured in twips. For example, if you set the TagWidth to 1000, the tag labels will always be 1000 twips wide.

Negative TagWidth values are interpreted as a percentage of the width of the control being labeled. For example, if you set TagWidth to 50, the tag labels will always be half as wide as the controls being labeled.

Setting TagWidth to zero disables the tag labels.

Label justification is determined by the CaptionPos property. The appearance and behavior of the tag labels is also affected by the AccessKey, WordWrap, and Font... properties.

Default Value Zero

Date Type Long

WordWrap Property [3.0]

Description This property determines whether the caption text should be automatically broken between words if a word would extend past the edge of the control. Return characters will also break the caption.

Usage [*form*.]*VSElastic*.**WordWrap** [= {**True**|**False**}]

Remarks If an Elastic has room for only one line of text and a caption with several words, word wrapping may hide parts of the caption. In such cases, setting this property to False may improve readability.

This property also affects the tag labels, if they are enabled (see the TagWidth property).

Default Value True

Date Type Boolean

IndexTab Reference

Description The VideoSoft IndexTab control allows you to group controls by subject, using the notebook metaphor that has become a Windows standard.

File Name VSVBX.VBX

Object Type VSIndexTab

Remarks To use the IndexTab control, follow these steps:

1. Draw the IndexTab.
2. Set the Caption property to create as many tabs as you need, separating the tabs with a pipe character (|).
3. Create one Picture Box or Elastic per tab, inside the IndexTab control.
4. Create the controls inside each container.
5. Set other IndexTab properties as desired.

Some applications need to change tab captions at run time. You can do this easily using the TabCaption property. For example, the following code changes the caption of the first tab in a IndexTab control:

```
VSIndexTab.TabCaption(0) = "new caption"
```

IndexTab Summary

Properties (default: CurrTab)

(About)	* AutoScroll	* AutoSwitch
BackColor	* BackSheets [3]	* BackTabColor
* BoldCurrent [4]	* BorderWidth [4]	BorderStyle
* Caption	* ClientLeft	* ClientHeight
* ClientTop	* ClientWidth	* CurrTab
* DogEars [4]	DragIcon	DragMode
Enabled	* FirstTab	FontBold
FontItalic	FontName	FontStrikethru
FontUnderline	ForeColor	* FrontTabColor
* FrontTabForeColor	Height	HelpContextID [3]
Hwnd [3]	Index	Left
* MouseOver [4]	MousePointer	* MultiRow [4]
Name	* NumTabs	* Position
* ShowFocusRect [3]	* Style	* TabCaption [3]
* TabColor [4]	* TabEnabled [4]	TabIndex
* TabOutlineColor [3]	* TabPicture [4]	* TabsPerPage [3]
TabStop	Tag	Top
Visible	Width	

Events

* Click	DblClick	DragDrop
DragOver	GotFocus	KeyDown
KeyPress	KeyUp	LostFocus
MouseDown	MouseMove	MouseUp
* Scroll		

Methods

Move	Refresh	SetFocus
ZOrder		

AutoScroll Property

Description Determines whether the control should automatically scroll the list of tabs so that the current tab is always visible.

Usage [*form.*]*VSIndexTab*.**AutoScroll** [= {**True**|**False**}]

Remarks If this property is set to True, the user can scroll the Tabs using the arrow keys, dog ears, or clicking on a partially hidden tab.

You can find out when scrolling happened by checking the Scroll event.

See also the DogEars property.

Default Value True

Date Type Boolean

AutoSwitch Property

Description Determines whether the control should automatically resize, show, and hide child windows when the current tab changes.

Usage [*form.*]*VSIndexTab*.**AutoSwitch** [= {**True**|**False**}]

Remarks To use AutoSwitch, you will normally create one container per tab. The IndexTab then takes care of switching and resizing the containers whenever a new tab is selected. To determine which container belongs to each tab, the IndexTab assumes that the leftmost container belongs to the first tab, the second from the left to the second tab, and so on.

You can also use the AutoSwitch feature at design time: double-click on any tab with the right mouse button. The tab will become active, and its container will be displayed just as it would at runtime. This makes form design easy.

To associate containers to tabs, right double-click on the current tab. This causes all tabs to be de-selected and the containers to be tiled in tab sequence. Reorganize the tabs by dragging them around until they are in the right sequence.

If AutoSwitch is on and CurrTab is set to a number greater than the number of containers, the last container is shown. This is useful if you have a single child in the IndexTab because it automatically sizes the child within the IndexTab.

Whenever the IndexTab switches the current tab, it disables all containers except the current one. This is done to prevent users from activating hidden controls with the keyboard and getting lost. If you need to enable and disable controls inside an IndexTab, you can either:

- Enable and disable the controls inside the containers, not the containers themselves, or
- Nest two containers. You can then enable and disable the inner one, and the IndexTab will take care of the outer one.

Default Value True

Date Type Boolean

BackSheets Property [3.0]

Description Defines the appearance of the back sheets behind the current tab.

Usage [*form.*]*VSIndexTab*.**BackSheets** [= *setting%*]

Remarks Valid options for this property are:

0 - None
1 - Shadow
2 - Sheets

This property is only active when the MultiRow property is set to False.

Default Value 2 - Sheets

Date Type Integer

BackTabColor Property

Description Defines the color of all non-selected tabs.

Usage [*form*.]*VSIndexTab*.**BackTabColor** [= *color&*]

Remarks Use this property in conjunction with BackColor and FrontTabColor to define the appearance of the control.

Default Value Light gray

Date Type Long

BoldCurrent Property [4.0]

Description Determines whether the current tab should be displayed with a bold font.

Usage [*form.*]*VSIndexTab*.**BoldCurrent** [= {**True|False**}]

Default Value True

Date Type Boolean

BorderWidth Property [4.0]

Description Sets and returns the width of the border between the IndexTab and its children. The border is measured in pixels and is in addition to the bevel drawn with the 3D styles.

Usage [*form.*]*VSIndexTab*.**BorderWidth** [= *setting%*]

Default Value 0

Date Type Integer

Caption Property

Description Determines the number of tabs and the text they contain.

Usage [*form*.]*VSIndexTab*.**Caption** [= *string$*]

Remarks Use the pipe character (|) to separate options. Use a dash (-) followed by spaces to create an invisible tab between regular tabs. Each tab may have an access key (&-key) to allow for fast tab switching.

Date Type String

Click Event

Description Fired when the current tab changes either through code, mouse, or keyboard action.

Syntax **Sub** *CtlName*_**Click ()**

ClientLeft, ClientTop, ClientWidth, ClientHeight Properties

Description Return the internal dimensions of the IndexTab excluding the rectangle used to show the tabs themselves. Units are twips.

Usage [*form*.]*VSIndexTab*.**ClientWidth**

Remarks These properties are useful if you want to size frames or picture controls so that they occupy the whole client area of the control.

You only need to use these properties if the AutoSwitch property is set to False.

These properties are read-only.

Date Type Single

CurrTab Property

Description Sets or returns the number of the currently selected tab.

Usage [*form*.]*VSIndexTab*.**CurrTab** [= *setting%*]

Remarks The first tab is numbered zero.

Setting this property to a negative value or to a value greater than the number of tabs causes all tabs to be de-selected. No error occurs.

Changing this property fires the Click event.

Default Value Zero

Date Type Integer

DogEars Property [4.0]

Description Determines whether the IndexTab should enable dog-ears, little flaps at the edges of the tab row that indicate there are more tabs off the screen.

Usage [*form*.]*VSIndexTab*.**DogEars** [= {**True|False**}]

Remarks This property only works when MultiRow is set to False. This is because when MultiRow is True, all tabs are automatically displayed and there is no need for scrolling.

Default Value True

Date Type Boolean

FirstTab Property

Description Sets or returns the number of the first tab to be displayed.

Usage [*form*.]*VSIndexTab*.**FirstTab** [= *setting%*]

Remarks The first tab is numbered zero. Invisible tabs, used to separate regular tabs, are also counted.

This property is useful if you want to implement scrolling in addition to that provided by the AutoScroll property.

Changes to this property fire the Scroll event.

Default Value Zero

Date Type Integer

FrontTabColor Property

Description Defines the color of the current tab and front page of the control.

Usage [*form*.]*VSIndexTab*.**FrontTabColor** [= *color&*]

Remarks Use this property in conjunction with BackColor and BackTabColor to define the appearance of the control.

Default Value White

Date Type Long

FrontTabForeColor Property [4.0]

Description Defines the color of the current tabs caption.

Usage [*form*.]*VSIndexTab*.**FrontTabForeColor** [= *color&*]

Remarks Use this property in conjunction with BackColor and BackTabColor to define the appearance of the control.

Default Value Black

Date Type Long

MouseOver Property [4.0]

Description Returns the number of the tab currently pointed to by the mouse, or -1 if the mouse is not over any tab.

Usage *variable%* = [*form.*]*VSIndexTab*.**MouseOver**

Remarks This property is read-only. It is especially useful when you want to provide dynamic help while the user moves the mouse over different tabs.

Date Type Integer

MultiRow Property [4.0]

Description Determines whether multiple tab rows should be displayed.

Usage [*form.*]*VSIndexTab*.**MultiRow** [= {**True**|**False**}]

Remarks If this property is set to False, tabs are allowed to spill off the screen and scroll back into view as needed. If it is set to True, multiple tab rows are created so that all tabs remain visible.

This property can only be used if TabsPerPage is set to a number greater than zero. When MultiRow is set to True, the BackSheets property is automatically disabled.

Default Value False

Date Type Boolean

NumTabs Property

Description Returns the number of tabs currently defined.

Usage [*form*.]*VSIndexTab*.**NumTabs**

Remarks This property is read-only. IndexTabs automatically sets it whenever you change the Caption or TabCaption properties.

Date Type Integer

Position Property

Description Sets or returns the position of the tabs within the control.

Usage [*form.*]*VSIndexTab*.**Position** [= *setting%*]

Remarks Valid options for this property are:

0 - Top
1 - Bottom
2 - Left
3 - Right
4 - Right, face in [3].

The difference between settings 3 and 4 is the text orientation. With setting 3, the text is written from the bottom up; with setting 4, the text is written from the top down (facing the inside of the IndexTab).

Windows only supports vertical text with some fonts. If you set the **Position** property to 2 or 3, the IndexTab will automatically convert the font into the nearest TrueType equivalent.

Default Value 1 - Bottom

Date Type Integer

Scroll Event

Description Fired when the **FirstTab** property changes either through code, mouse, or keyboard action.

Syntax **Sub** *CtlName*_**Scroll ()**

ShowFocusRect Property

Description Determines whether the control should show a focus rectangle around the current tab when it has the focus.

Usage [*form.*]*VSIndexTab*.**ShowFocusRect** [= {**True**|**False**}]

Default Value True

Date Type Boolean

Style Property

Description Sets or returns the style used to draw the tabs.

Usage [*form.*]*VSIndexTab*.**Style** [= *setting%*]

Remarks Valid options for this property are:

0 - Slanted lines
1 - Slanted lines plus 3D effect
2 - Rounded corners
3 - Rounded corners plus 3D effect
4 - Chamfered corners
5 - Chamfered corners plus 3D effect
6 - Straight [4.0]
7 - Straight plus 3D effect [4.0]

Default Value 0 - Slanted Lines

Date Type Integer

TabCaption Property [3.0]

Description Sets or retrieves the caption for an individual tab.

Usage [*form.*]*VSIndexTab*.**TabCaption(***index%***)** [= *string$*]

Remarks Use this property to dynamically change tabs in the IndexTab.

You can create new tabs by assigning a string with embedded pipe characters to a TabCaption. You can delete tabs by assigning them an empty string.

Trying to access a tab smaller than 0 or greater than **NumTabs**-1 will generate an invalid index error.

This property is not available at design-time.

Example

```
VSIndexTab.Caption = "First|Second|Third"
debug.print VSIndexTab.TabCaption(0)
 First

VSIndexTab.TabCaption(0) = "First|New Second"
debug.print VSIndexTab.Caption
 First|New Second|Second|Third

VSIndexTab.TabCaption(0) = ""
debug.print VSIndexTab.Caption
 New Second|Second|Third
```

Date Type String array

TabColor Property [4.0]

Description Sets or retrieves the color for an individual tab.

Usage [*form*.]*VSIndexTab*.**TabColor(***index%***)** [= *colorref&*]

Remarks Trying to access a tab smaller than 0 or greater than NumTabs-1 will generate an invalid index error.

This property is not available at design-time.

Date Type Long array

TabEnabled Property [4.0]

Description Sets or retrieves the enabled status for an individual tab.

Usage [*form*.]*VSIndexTab*.**TabEnabled(***index%***)** [= {**True**|**False**}]

Remarks Disabled tab captions are displayed in light gray. The user cannot activate them with the mouse or keyboard.

This property is not available at design-time.

Date Type Boolean array

TabOutlineColor Property [3.0]

Description Defines the color to be used for the outline of the tabs and sheets.

Usage [*form*.]*VSIndexTab*.**TabOutlineColor** [= *color&*]

Remarks When using one of the 3D styles, set this property to black to get the best results.

Default Value Black

Date Type Long

TabPicture Property [4.0]

Description Allows you to specify pictures to be shown within individual tabs.

Usage [*form*.]*VSIndexTab*.**TabPicture(***index%***)** [= *pictureref%*]

Remarks You can assign bitmaps or icons to tabs (metafiles are not allowed). After each picture is assigned to a tab, the control automatically resizes to fit the largest picture.

This property is not available at design-time.

Date Type Picture array

TabsPerPage Property [3.0]

Description Determines the number of tabs that should fit across the control.

Usage [*form.*]*VSIndexTab*.**TabsPerPage** [= *n%*]

Remarks If this property is set to zero, the tabs are drawn as close together as possible, and each tab may have a different size.

If this property is set to a positive number *n%*, then *n%* tabs are shown on the page, all with the same size.

If MultiRow is set to true, this property must be set to a number greater than zero.

Default Value Zero

Date Type Integer

Awk Reference

Description The VideoSoft Awk control allows you to quickly scan and parse text files. It is the ideal tool for doing simple data manipulation changing its format, checking its validity, retrieving items, generating reports, and the like without having to write a lot of code to scan and parse files.

File Name VSVBX.VBX

Object Type VSAwk

Remarks The VideoSoft Awk control is similar to the original AWK language in concept and purpose, but it is substantially different in syntax, since the VSVBX Awk is embedded in Visual Basic. The most apparent omission is a pattern matching mechanism, which is provided by VBs **Like** operator.

Although Awk was designed primarily for scanning and parsing files, you can use it to parse anything you want. For example, the following routine parses a file specification into drive, path, and file name:

```
Sub ParseFileSpec (FileSpec as String)

  ' set line and field separator properties
  VSAwk = FileSpec
  VSAwk.FS = "\"

  ' get the file name and clear its field
  Debug.Print "file "; VSAwk.F(VSAwk.NF)
  VSAwk.F(VSAwk.NF) = ""

  ' get the drive and creal its field
  Debug.Print "drive "; VSAwk.F(1)
  VSAwk.F(1) = ""

  ' whatever is left is the dir
  Debug.Print "dir "; VSAwk
End Sub
```

Awk can read lines up to 64k characters long while scanning files. If a line exceeds this limit, Awk truncates the line and generates an **Error** event.

Awk is different from most other custom controls in that it does not generate run time errors. Instead, it fires the **Error** event and sets its own internal **Error** property. Therefore, Awk ignores errors by default. You can trap errors either by trapping the Error event or by testing the Error property immediately after using any other Awk property.

Awk Summary

Properties (default: L)

(About)	* Action	* Case
* CurrPos	* Error	* F
* FieldAt	* FileName	* FileType
* FilterQuotes [4]	* FS	Index
* L	* MatchQuotes [3]	Name
* NF	* PercentDone	* PosAt [3]
* RN	Tag	* Val [4]

Events

* Begin	* End	* Error
* Scan	* Variable [4]	

Action Property

Description Specifies an action to be performed by the Awk control.

Usage [*form.*]*VSAwk*.**Action** [= *action%*]

Remarks Valid actions are:

0 - Scan File
1 - Next Line
2 - Close File

The *Scan File* action causes Awk to take the following actions:

1. Open the file specified by the FileName property;
2. Fire the Begin event;
3. Read the file parsing each line and firing a Scan event for each one;
4. Close the file;
5. Fire the End event.

The *Next Line* action causes Awk to read the next line from the current data file. It can be useful in the routine that handles the Scan event.

The *Close File* action causes Awk to close the current file and stop scanning it.

Date Type Integer

Begin Event

Description Fired when Awk opens a file for scanning, after Action is set to 0 (Scan File).

Syntax **Sub** *CtlName*_**Begin ()**

Case Property

Description Determines whether Awk should change the case of each line as it is read.

Usage [*form.*]*VSAwk*.**Case** [= *setting%*]

Remarks Valid setting for this property are:

0 - No Change
1 - Convert to Upper Case
2 - Convert to Lower Case

Default Value 0 - No Change

Date Type Integer

CurrPos Property

Description Sets or returns the position of the current line within the file.

Usage [*form*.]*VSAwk*.**CurrPos** [= *value&*]

Remarks This property is useful when you want to save the position of current line so that you can quickly return to it later.

Date Type Long

End Event

Description Fired when Awk is done scanning a file.

Syntax **Sub** *CtlName*_**End ()**

Error Event

Description Fired when Awk encounters an error. See the Error property for a list of possible errors.

Syntax **Sub** *CtlName*_**Error ()**

Error Property

Description Returns a code that identifies the last error trapped by Awk.

Usage [*form.*]*VSAwk*.**Error**

Remarks Possible error codes are:

Parser errors:
0 - No Error
1 - Can't open file
2 - Line too long
3 - Field too long
4 - Invalid field index set
5 - Invalid field index get

Expression evaluator errors:
6 - Syntax
7 - Bad Token
8 - Unbalanced Brackets
9 - Division by Zero
10 - Argument domain error
11 - Argument singularity
12 - Overflow range error
13 - Partial loss of significance
14 - Total loss of significance
15 - The result is too small to be represented
16 - Not Reentrant (see the Variable event for explanation)

The *Can't open file* error occurs when the file specified by the FileName property can't be found.

The *Line too long* error occurs when the file contains lines longer than 64k characters. In this case, Awk truncates the line.

The *Field too long* error occurs when you try to make the current line longer than 64k characters.

The *Invalid field index set* error occurs when you try to set the value of a field that doesn't exist, i.e., smaller than zero or greater than NF.

The *Invalid field index get* error occurs when you try to get the value of a field that doesn't exist, i.e., smaller than zero or greater than NF.

The expression evaluator errors occur when you try to evaluate an expression using the Val property and the expression is invalid.

Date Type Integer

F (Field) Property

Description Sets or returns the value of a field within the current line (record).

Usage [*form.*]*VSAwk*.**F(***index%***)** [= *value$*]

Remarks Fields are numbered from 1 to NF. As in the Awk language, field 0 corresponds to the whole line (record).

If you try to set or read an invalid field, Awk fires the Error event.

If you set a field to a string that contains field separators, the line is automatically re-parsed, and the NF property is updated.

Example

```
VSAwk.F(0) = "This is a string"
Debug.Print VSAwk.NF, VSAwk.F(0)
 4    This is a string

Debug.Print VSAwk.F(4)
 string

VSAwk.F(4) = "plain long ASCII string"
Debug.Print VSAwk.NF, VSAwk.F(0)
 7    This is a plain long ASCII string
```

Date Type String array.

FieldAt Property

Description Returns the number of the field at the specified string position.

Usage [*form.*]*VSAwk*.**FieldAt(***position%***)**

Remarks This property was designed to be used with VBs **InStr** function. The first string position is 1.

This property is useful when you want to process a field based on its contents rather than on its position. See also the PosAt property, which performs the reverse function.

Example Use the PosAt property to find the contents of the field after the string value.

```
VSAwk = "stuff stuff stuff stuff value data stuff stuff"
debug.print VSAwk.FieldAt(instr(VSAwk, "value"))
 5

debug.print VSAwk.f(5+1)
 data
```

Date Type Integer array.

FileName Property

Description Sets or returns the name of the file to be scanned by the Awk control.

Usage [*form.*]*VSAwk*.**FileName** [= *filename$*]

Remarks If Awk can't find the specified file, it fires the Error event.

Date Type String.

FileType Property

Description Determines whether the file being scanned is a text or binary file. When reading text files, Awk translates carriage return/line feed characters into carriage returns and closes the file when an end-of-file character (ctrl-Z) character is found.

Usage [*form*.]*VSAwk*.**FileType** [= *setting%*]

Remarks Valid settings for this property are:

0 - Text File
1 - Binary File

Default Value 0 - Text File.

Date Type Integer

FilterQuotes Property [4.0]

Description Determines whether should remove leading and trailing quotes from fields when the **MatchQuotes** property is set to True.

Usage [*form.*]*VSAwk*.**FilterQuotes** [= {**True|False**}]

Remarks This property has no effect when **MatchQuotes** is set to False.

Example

```
VSAwk1 = "Name 'John McAdam' Age 23"
VSAwk1.MatchQuotes = True

VSAwk1.FilterQuotes = False
debug.print VSAwk1.F(2)
  'John McAdam'

VSAwk1.FilterQuotes = True
debug.print VSAwk1.F(2)
  John McAdam
```

Default Value False.

Date Type Boolean.

FS (Field Separator) Property

Description Sets or returns the string of characters to be used by Awk as field separators.

Usage [*form.*]*VSAwk*.**FS** [= *string$*]

Remarks Each occurrence of any of the characters in the FS string marks the beginning of a new field. The only exception is that repeated spaces are equivalent to a single space. See the examples below.

If you change this property, the current string is automatically re-parsed.

Example

```
VSAwk = "This is a string|||with some      pipes in it"
VSAwk.FS = " "
debug.print VSAwk.NF
 8

VSAwk.FS = "|"
debug.print VSAwk.NF
 4

VSAwk.FS = "| "
debug.print VSAwk.NF
 11
```

Default Value , (space, comma, tab)

Date Type String.

L (Line) Property

Description Sets or returns the value of the current line (record).

Usage [*form.*]*VSAwk*.**L** [= *string$*]

Remarks This is the default property. This property is exactly the same as the F(0) property.

Example

```
VSAwk.F(0) = "This is a string"
Debug.Print VSAwk.L
 This is a string
```

Date Type String.

MatchQuotes Property [3.0]

Description Determines whether the Awk parser should regard quote-delimited strings as single fields.

Usage [*form.*]*VSAwk*.**MatchQuotes** [= {**True**|**False**}]

Remarks This feature was added to facilitate parsing of quote and comma-delimited files such as those exported by most database and spreadsheet programs.

Double or single quotes are both recognized, but one does not match the other. Unmatched quotes in a line are ignored.

Example

```
VSAwk = "'John Doe', 24, '1244 Main Street', 645-5432"
VSAwk.FS = ", "
VSAwk.MatchQuotes = false
debug.print VSAwk.NF, VSAwk.F(1)
 7   'John

VSAwk.MatchQuotes = true
debug.print VSAwk.NF, VSAwk.F(1)
 4   'John Doe'
```

Default Value False

Date Type Boolean

NF (Number of Fields) Property

Description Returns the number of fields in the current line (record).

Usage [*form*.]*VSAwk*.**NF**

Remarks The number of fields changes whenever the current line (L property) or field separators (FS property) change.

Date Type Integer

PercentDone Property

Description Returns a number between 0 and 100 that indicates how much of the current file has been scanned.

Usage [*form*.]*VSAwk*.**PercentDone**

Remarks This property was designed to give you an easy way to report progress while processing long files. It can be used, for example, with the Elastic FloodPercent property.

Date Type Integer

PosAt Property [3.0]

Description Returns the position of the specified field in the current record.

Usage [*form.*]*VSAwk*.**PosAt(***field%***)**

Remarks This property was designed to be used with VBs string functions. The first string position is 1.

This property is useful when you want to process a field based on its contents rather than on its position. See also the FieldAt property, which performs the reverse function.

Example Use the PosAt property to remove the first four fields from a record.

```
VSAwk = "junk1 junk2 junk3 junk4 data5 data6 data7"
debug.print VSAwk.PosAt(5)
 25

debug.print mid(VSAwk,25)
 data5 data6 data7
```

Date Type Integer array.

RN (Record Number) Property

Description Returns the number of the current line (record) being scanned by Awk.

Usage [*form*.]*VSAwk*.**RN**

Date Type Long

Scan Event

Description Fired for each line in the input file while Awk scans a file. This is where you put the code that does the actual processing.

You may close the current file or get the next input line while in this subroutine by setting the Action property to 2 or 3, respectively.

Syntax **Sub** *CtlName*_**Scan ()**

Val Property [4.0]

Description Evaluates the mathematical expression in the Caption property and returns its value. The expression may contain variables.

Usage *variable!* = [*form.*]*VSAwk*.**Val**

Remarks The following operators are supported, according to their priority:

Prty	Op	Description	Op	Description
6	**()**	subexpressions		
5	**^**	power		
4	*****	times	**/**	divide
	%	modulus	****	integer divide
3	**+**	plus	**-**	minus
2	**>**	greater than	**>=**	greater or equal
	<	less than	**<=**	less or equal
	=	equal	**<>**	not equal
1	**&**	logical and	**|**	logical or

The following built-in functions are supported (they are case-insensitive): abs, sin, cos, tan, atn, log, and exp.

The Awk expression allows you to define your own variables. This is done in a very simple way: whenever the parser encounters a string that is not a number, operator, or built-in function, it fires the Variable event. The event handler gets the variable name and is responsible for returning its value. For more details on this feature, see the *Variable* event.

If an error occurs while evaluating an expression, the Error property is set to one of the codes listed below. *Always* check the Error property after using Val.

6 - Syntax
7 - Bad Token
8 - Unbalanced Brackets
9 - Division by Zero
10 - Not Reentrant (see the Variable event for explanation)

Example:

```
VSAwk1 = "(1+2)*3^2"
debug.print VSAwk1.Val
 27
```

Date Type Single

Variable Event [4.0]

Description Fired when you read the Val property to evaluate an expression and the expression contains variables. Use this event to supply the variable values.

Syntax **Sub** *CtlName*_**Variable (***Variable As String, Value As Single, Accept As Integer***)**

Variable: Name of the variable that needs to be evaluated
Value: Value of the variable
Accept: Flag that indicates the variable is valid

Remarks Your event handler will typically read the variable name from the *Variable* parameter, supply its value through the *Value* parameter, and set the *Accept* flag to indicate the variable is valid. If the variable name is not valid, return without setting the *Accept* parameter and Awk will generate a Bad Token error.

You may not use the Val property while handling the Variable event. Doing so will trigger an error.

Example If your program defines two variables x and y, your event handler might look like this:

```
Sub VSAwk1_Variable (Variable$, Value!, Accept%)
  Accept% = True
  Select Case Variable$
    Case "X", "x"
      Value! = GlobalX!
    Case "Y", "y"
      Value! = GlobalY!
    Case Else
      Accept% = False ' unrecognized variable will trigger error
  End Select
End Sub
```

Troubleshooting

This section contains answers to the most common questions people ask our technical support staff. Read it even if you havent run across any problems you may find some useful tips in here.

Trouble Invisible controls do not resize in the right place when made visible.

Explanation When resizing controls, the Elastic ignores invisible controls (you'd get gaps if it didn't). When they become visible again, they appear and are resized normally. If the form has been resized in the meantime, however, all the other controls have been moved, except the invisible ones. Thats why they may not appear in the right position.

Solution Before changing the visibility of the child controls, rearrange their positions with code. For example, if you want to make a control named *foo* visible and you want it to appear next to a control named *bar*, use the following code:

```
foo.left = bar.left + 10 ' (assuming even horizontal spacing)
foo.visible = True
```

Trouble Some controls cause flicker when resized by the Elastic (e.g. ComboBoxes).

Explanation Some controls, such as ComboBoxes, cannot be freely resized. When you resize a ComboBox with the mouse, for example, it quickly snaps back to its old height. The same happens when the Elastic tries to resize it, and that results in some flicker.

Other controls are graphical (e.g. labels and image boxes). Graphical controls save some resources, at the expense of less efficient redrawing. These controls can cause some flicker when they are resized by the Elastic.

Solution Design your forms normally. If you get a lot of flicker when resizing, try the following:

1 - Make the Form and the Elastics background color the same. This often makes resizing seem faster.

2 - Set the Elastics BevelInner property to 0 - None. This can significantly speed up repainting.

3 - Replace graphical controls with non-graphical ones (elastics instead of labels, picture boxes instead of image boxes).

4 - Reduce the number of controls inside the Elastic (use tag labels, or an IndexTab to reduce clutter).

Trouble The IndexTab changes the Enabled property of its child controls.

Explanation Whenever the IndexTab switches the current tab, it disables all containers except the current one. This is done to prevent users from activating hidden controls with the keyboard and getting lost.

Solution There are two easy solutions for this:

1 - Do not enable and disable the containers, only the controls inside them, which the IndexTab does not touch.

2 - If you have too many controls to enable and disable, use two nested containers. You can then enable and disable the inner one, and the IndexTab will take care of the outer one.

Order Form

TO: VideoSoft
2625 Alcatraz Avenue, Suite 271
Berkeley, California 94705

(To order by phone, call 1-800-5477295)

Please register my copy of the VideoSoft Custom Control Library. I am enclosing a check or money order for the amount of:

Single developer	US$45.00
Additional developers (see note below): _ x 45 =	
Shipping and Handling Domestic	6.00
Shipping and Handling International	10.00
8.5% tax (CA residents only)	
TOTAL	

Note: You may take advantage of VideoSofts ***Power Of 2*** site licenses: Buy ***n*** licenses and install the VBX and developers license files on ***n2*** computers within the same corporation. (Of course, you may still distribute the VBX without the developers license file to all your clients, royalty-free.)
For example: If you buy 2 VSVBX licenses, you will receive 2 sets of manuals and disks, but you may install the product on 4 computers.

Name:

Company:

Street:

City, State, ZIP:

Country:

Phone:

Where did you hear about the VideoSoft Custom Controls?