VSOCX/VSVBX 5.0

VideoSoft Custom Control Library

To learn how to use help, press F1

To learn how to use VSOCX or VSVBX, follow these steps:

1 - Run the sample programs and study the source code provided,
2 - Follow the VSOCX/VSVBX QuickStart tutorial,
3 - Use this help file as a reference.

Introduction

Find out about installation, product support, licensing, registration., and other VideoSoft products.

QuickStart

Follow step-by-step instructions on how to create and use the VSVBX controls.

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.

vsIndexTab

Allows you to group controls by subject, using the familiar notebook metaphor that has become a Windows standard.

vsAwk

Parsing engine named and patterned after the popular Unix utility, plus a powerful expression evaluator.

Introduction

Welcome to VSVBX 5.0, a VideoSoft Custom Control Library. If you are upgrading from a previous version, make sure to look in the *New Features in VSVBX 5.0* section.

This release includes two versions of the controls: the original VSVBX and the new VSOCX. We worked very hard to make both versions totally compatible, so all the information in this document applies to both versions, except when explicitely stated.

The VSVBX controls are designed to save you from writing tedious, repetitive, error-prone code. 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) 704-8200 (phone)
(510) 843-0174 (fax)

Installation

To install the OCX version, VSOCX, use the INSTALL utility provided on the distribution diskette.

To install the VBX version, 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 VSVBX license file. If VSVBX cannot find this file when it starts running, it displays a VideoSoft banner and waits for the user to click Ok, unless the project was compiled on a machine that had VSVBX.LIC installed.

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

Distribution

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

You cannot distribute the license file VSVBX.LIC. And you dont have to: as long as you have the license file installed on your machine, VSVBX will stamp every application you compile so the banner will not appear when your users run the applications.

If you work with other developers, you may be interested in VideoSofts site licenses. Call us for details.

If you havent yet registered your copy of VSVBX and would like to do it now, click **HERE** to get a Registration Form.

Product Support

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

Internet	http:\\www.videosoft.com
CompuServe	CIS 74774,420 or join our forum by typing **GO VIDEOSOFT**
Mail	VideoSoft 2625 Alcatraz Avenue, Suite 271 Berkeley, California 94705
Phone	(510) 704-8200
Fax	(510) 843-0174

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.

Also, please make sure you check the last section of the manual, Hints and Troubleshooting. It contains answers to the most common questions people ask our technical support staff. Maybe you can find your answer there.

Differences Between the OCX and VBX Versions

The main differences between the VBX and OCX versions of the VSVBX controls are summarized below:

- **Installation:**
  The VBX version may be installed manually, as described in the *Installation* seciton of this document. The OCX should be installed with the supplied install program, which copies all files required and also takes care of registering the controls in the systems registry.

- **IndexTab default style:**
  The default style for new IndexTab controls is different for the VBX and OCX versions. For the VBX version, the style is flat, with tabs aligned at the bottom of the control. For the OCX version, the defatult style is similar to the Win32 property sheets.

- **IndexTab design-time behavior:**
  The command to switch tabs at design-time, is different for the VBX and OCX versions. For the VBX version, you must double-click on the new tab with the right mouse button. For the OCX version, you must click on the new tab with the left mouse button.

- **Awk Value property type and Variable event:**
  Both the *Val* property and the *Value* parameter in the Awks *Variable* event are of type Single in the VBX version and Double in the OCX version. If you are migrating projects from the VBX to the OCX versions and using these properties/events, you will have the chance to modify the types of the variables involved and the declaration for the *Variable* event as follows:

```
'
' Original code (VBX version)
'
Sub vsAwk1_Variable(Variable As String,
                    Value As Single,
                    Accept As Integer)

'
' New code (OCX version)
'
Private Sub vsAwk1_Variable(Variable As String,
                            Value As Double,
                            Accept As Integer)
```

New Features in VSVBX 5.0

This section summarizes the new features in VSVBX 5.0. If you are familiar with earlier versions of VSVBX, this section will get you up to speed quickly. For details on each new feature, check the main body of the documentation.

VSVBX 5.0 has several significant improvement over earlier versions. First of all, it works with C, C++, and all other environments that support the VBX level 1.0 protocol. VSVBX 5.0 also supports versioning, including the standard version information for use with VER.DLL and a new Version property for all controls.

In response to requests from many users, VSVBX 5.0 requires only one license file (VSVBX.LIC) on the developers machine. See the Installation and Distribution sections for details.

VSVBX 5.0 is easier to use, faster, better-looking, and it offers 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.

Most of these enhancements were added in response to user requests, and we are grateful for their input.

Elastic 5.0

The new Elastic has 11 new properties, as well as new settings for several of the older properties. The new Elastic offers a better design-time environment, faster repainting, new frame styles, and pictures.

New Properties

FreezeElastics is a design-time property that temporarily disables all Elastic aligning. This lets you move Elastics around your forms easily.

Template is another design-time property. It allows you to pick from a list of predefined styles and automatically sets a bunch of properties to their appropriate values. Its a big time-saver.

Picture and **PicturePos** allow you to place any type of picture anywhere on the Elastic. Its more flexible than using Picture or Image controls.

CaptionStyle allows you to specify 3-D effects for the Caption and tag label text. Several styles are available.

ButtonState and **CornerColor** work with the new settings of the Style property: *Command Button* and *Radio Button*. You can tell the Elastic to behave like a button, complete with captions, pictures, accelerator keys, and even child controls.

BevelOuterDir allows you to specify whether the Elastics outer bevels should be drawn in both directions, only horizontally, or only vertically. This makes it easy to create great-looking tool bars and gives you flexibility to create new effects.

QuickPaint is similar to the AutoRedraw property of Picture boxes. It causes the Elastic to draw itself in memory, and then copy itself to the screen when its ready. This process reduces flicker.

TagSplit works in conjunction with the Elastics tag labels. If you set TagSplit to True, then the Elastic splits child control Tags at the first pipe character it finds, and uses only the beginning of the caption to label to controls. You can use remaining portion of the tag for whatever purposes you want.

ForeColorDisabled allows you to specify a color to be used when drawing the caption and tag labels when the Elastic is disabled.

SetParent gives you a convenient way to add children to an Elastic at run-time.

Version allows you to determine what version of the VBX is loaded while your program is executing. If you detect an old version, you may disable some features of your code or warn the user to get an upgrade.

New Settings

The **Style** property has the following new settings: *3 - Command Button*, and *4 Radio Button*. Use them to create buttons with pictures, wrapping captions, accelerator keys, and even child controls.

The **TagPosition** property has the following new settings: *2 Right* and *3 Below.* Now you have total flexibility to control the position of the tag labels.

The **BevelInner** and **BevelOuter** properties have the following new settings: *8 - Raised New Look* and *9 - Inset New Look*. These settings give your application the new 3-D look familiar to Visual C++ and Windows 95 users.

IndexTab 5.0

The new IndexTab has a new event and 5 new properties, as well as new settings for several of the older properties.

New event

Switch is an event fired right before the currant tab changes. This allows you to save information on the current tab page before the user switches to a different page, or even to cancel the switch. Unlike the Click event, Switch only gets fired when you are actually changing tabs.

New properties

TabHeight and **MultiRowOffset** give you additional control over the look of your tabs.

Template is a design-time property that allows you to pick from a list of predefined styles and automatically sets a bunch of properties to their appropriate values. Pick a look, and let the IndexTab configure itself.

CaptionStyle allows you to use 3-D captions for your tabs. A variety of effects are available.

New3D allows you to give your tabs the new 3-D look familiar to Visual C++ and Windows 95 users.

GetTab allows you to determine the tab page on which a given control is.

SetParent gives you a convenient way to add pages to an IndexTab at run-time.

Version allows you to determine what version of the VBX is loaded while your program is executing. If you detect an old version, you may disable some features of your code or warn the user to get an upgrade.

New Settings

The **Style** property has two new settings: *8 CutCorners* and *9 CutCorners 3D*.

The **TabPicture** property now supports metafiles.

Awk 5.0

The new Awk has two new properties and a couple of new functions added to its expression evaluator. The new functions are: SQRT, INT, FRAC, CEIL, and FLOOR. See the **Val** property for details.

New property

FindLine allows you to scan a file for a line containing a given string.

Version allows you to determine what version of the VBX is loaded while your program is executing. If you detect an old version, you may disable some features of your code or warn the user to get an upgrade.

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 VSOCX16.OCX, VSOCX32.OCX or VSVBX.VBX

Object Type vsElastic

Note Before you can use a vsElastic control in your application, you must add VSOCX/VSVBX to your project (see the Visual Basic manual for details). To automatically include VSVBX/VSOCX in new projects, put it in an AUTOLOAD file. When distributing your application, you should install the VSVBX/VSOCX files in the user's Microsoft Windows SYSTEM subdirectory.

Remarks 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
* BevelOuterDir [5]	* BevelOuterWidth	* BorderWidth
* ButtonState [5]	Caption	* CaptionPos
* CaptionStyle [5]	* ChildSpacing	* CornerColor [5]
DragIcon	DragMode	Enabled
* FloodColor	* FloodDirection	* FloodPercent
FontBold	FontItalic	FontName
FontSize	FontStrikethru	FontUnderline
ForeColor	* ForeColorDisabled [5]	* FreezeElastics [5]
Height	HelpContextID [3]	Hwnd [3]
Index	Left	* MaxChildSize [3]
* MinChildSize [3]	MousePointer	Name
Parent	* Picture [5]	* PicturePos [5]
* QuickPaint [5]	* SetParent [5]	* ShadowColor [4]
* ShowFocusRect [5]	* ShowOutline [5]	* Splitter
* Style [4]	TabIndex	TabStop
Tag	* TagPosition [4]	* TagSplit [5]
* TagWidth [3]	* Template [5]	Top
Visible	* Version [5]	Width
* WordWrap [3]		

Events

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

AccessKey Property

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 select controls as if the Elastic were a label.

If tag labels are enabled (see the TagWidth property) each child control tag may contain a separate access key. When an access key is pressed, the elastic will choose the corresponding child control and pass it the focus.

If the Elastic is configured as a button, setting AccessKey to True will cause the button to be clicked when the access key is pressed.

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]
6 - Elastics Vertically [3]
7 - Proportional [3]

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.*

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]
6 - Groove [4]
7 - Shadow [4] (available for **BevelInner** only)
8 - Raised New Look [5]
9 - Inset New Look [5]

The *New Look* styles give your controls a contemporary 3D look. They always have width 2, regardless of the setting of the BevelWidth property.

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]

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

BevelOuterDir Property [5]

Description This property allows you to specify whether the Elastics outer bevel should be applied in both directions, only to the horizontal (along the top and bottom edges of the control), or only to the vertical direction (along the left and right edges of the control).

Usage *[form!]vsElastic*.**BevelOuterDir** [= *setting%*]

Remarks Valid settings for this property are:

0 - Both
1 - Horizontal
2 - Vertical

This property is mainly useful when you want to create toolbars or forms with a crease along one direction.

Default Value 0 - Both

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

ButtonState Property [5]

Description This property sets or returns the state of an Elastic configured as a Toggle Button.

Usage *[form!]vsElastic*.**ButtonState** [= *setting%*]

Remarks This property only works when the Elastics Style property is set to *4 Toggle Button*.

Data Type Boolean

CaptionPos Property

Description Sets or returns the position of the Elastics caption. Also determines 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

CaptionStyle Property [5]

Description Allows you to specify a 3D effect for the Elastics caption.

Usage *[form!]vsElastic*.**CaptionStyle** [= *setting%*]

Remarks Valid settings for this property are:

0 - Flat
1 - Raised
2 - Inset
3 - Raised Light
4 - Inset Light

Settings 1 and 2 work best for large and bold fonts. Settings 3 and 4 work best for small regular fonts.

Default Value 0 - Flat

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

CornerColor Property [5]

Description Allows you to specify the color to be used when painting the Elastics corners.

Usage *[form!]vsElastic*.**CornerColor** [= *color&*]

Remarks If you set this property to the background color of the Elastics parent, the Elastic will have a rounded look. This is mainly useful if you want to use Elastics as buttons.

If you set this property to 0, the Elastic ignores it and paints the corners as if they were part of the outer bevel.

Default Value 0

Data Type Long (color)

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 (color)

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

Data Type Integer

ForeColorDisabled Property [5]

Description This property allows you to specify a color to be used for the Elastics caption and tag labels when the Elastic is disabled.

Usage *[form!]vsElastic*.**ForeColorDisabled** [= *color&*]

Remarks If you set this property to 0, the Elastic ignores it and always uses the ForeColor property for the caption and tag labels.

Default Value 0

Data Type Long (color)

FreezeElastics Property [5]

Description This property allows you to temporarily disable the automatic alignment feature for *all* Elastics on all forms.

Usage *[form!]vsElastic*.**FreezeElastics** [= {**True**|**False**}]

Remarks If you set Freeze Elastics to True, all Elastics will stop resizing and realigning themselves automatically. This can make it easier for you to select them with the mouse and arrange them on the form.

The value of this property is shared by *all* Elastics. When you set FreezeElastics to True, all Elastics will temporarily stop resizing and realigning themselves. When you set it to False, all Elastics will resume their normal behavior.

This property is available at design-time only.

Default Value False

Data Type Boolean

MaxChildSize Property [3]

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

Data Type Long

MinChildSize Property [3]

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

Data Type Long

Picture Property [5]

Description This property allows you to specify a picture to be painted inside the Elastic.

Usage *[form!]vsElastic*.**Picture** [= *picture%*]

Remarks The picture is always painted inside the Elastics borders. The position of the picture is determined by the PicturePos property.

The Elastic can display bitmaps, icons, and metafiles.

Data Type Picture

PicturePos Property [5]

Description This property sets or returns the position where the Elastics picture will be displayed.

Usage *[form!]vsElastic*.**PicturePos** [= *setting%*]

Remarks Valid settings 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
9 - Stretch

All Elastic pictures are drawn within the Elastics borders.

Metafiles are always stretched to fill all the space available.

Default Value 4 - Center Center

Data Type Integer

QuickPaint Property [5]

Description Setting this property to True causes the Elastic to draw itself into memory and then copy the new image to the screen at once. This eliminates flicker.

Usage *[form!]vsElastic*.**QuickPaint** [= {**True**|**False**}]

Remarks This property is similar to the Picture controls AutoRedraw feature. When it is set to True, the Elastic needs some memory and some time to generate its image, but repaints often appear much faster and flicker-free since the image is copied to the screen at once.

This option is especially useful if you change tag labels dynamically.

This is similar to the Picture control's AutoRedraw property.

Default Value False

Data Type Boolean

RealignFrame Event

Description Fired after the Elastic is resized.

Syntax **Sub** *vsElastic*_**RealignFrame ()**

ResizeChildren Event

Description Fired after the Elastics children are resized.

Syntax **Sub** *vsElastic_***ResizeChildren ()**

SetParent Property [5]

Description Allows you to add children to an Elastic at run time.

Usage *[form!]vsElastic*.**SetParent** = *NewChild.Hwnd*

Remarks This property is rarely needed with Visual Basic because you can create child control by drawing them directly over the Elastic.

Setting this property to an invalid HwnD value will trigger a run-time error. Invalid HwnD values are those that do not correspond to a window, those of windows that contain the Elastic, and those of top-level windows (forms).

You cannot remove children using this property. If you need to do that, declare and use the Windows API SetParent function.

This property is write-only.

Data Type Integer (Hwnd)

ShadowColor Property [4]

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** [= *color&*]

Default Value Dark Gray

Data Type Long (color)

ShowFocusRect Property [5]

Description Determines whether the Elastic should show a focus rectangle when it has the focus, like standard command buttons do.

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

Remarks The Elastic will only get the focus if its Style property is set to one of the button styles and the TabStop property is set to True.

The focus rectangle will be drawn around the caption, if there is one, or around the Elastics border, if there is no caption.

Default Value False

Data Type Boolean

ShowOutline Property [5]

Description Determines whether the Elastic should display a black outline when it has the focus, like standard command buttons do.

Usage *[form!]vsElastic*.**ShowOutline** [= {**True**|**False**}]

Remarks The Elastic will only get the focus if its Style property is set to one of the button styles and the TabStop property is set to True.

The focus rectangle will be drawn around the caption, if there is one, or around the Elastics border, if there is no caption.

Default Value False

Data Type Boolean

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 **AutoSizeChildren** property must be set to *Uneven Vertical* or *Uneven Horizontal*.

Default Value False

Data Type Boolean

Style Property [4]

Description Determines whether the Caption should be positioned in the Elastics client area or over its frame, similar to the standard frame control. Also allows you to use the Elastic as a button.

Usage *[form!]vsElastic*.**Style** [= *setting%*]

Remarks Valid settings for this property are:

0 - Classic
1 - Frame Top
2 - Frame Bottom
3 - Command Button [5]
4 - Toggle Button [5]

The difference between the button styles is that the command button is a regular push button, while the Toggle Button is a toggle. The state of the Toggle Button can be set or retrieved through the ButtonState property.

When you set the Style property to 3 or 4 (the button styles) and the TabStop property to True, the Elastic becomes able to gain the focus. If the ShowFocusRect property is set to True, the Elastic will also draw a focus rectangle when it does gain the focus.

Default Value 0 - Classic

Data Type Integer

TagPosition Property [4]

Description Determines where the tag labels should be displayed with respect to 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
1 - Above
2 - Right [5]
3 - Below [5]

Default Value 0 - Left

Data Type Integer

TagSplit Property [5]

Description Determines whether the Elastic should use the whole Tag property of its children when painting the tag labels or only the portion up to the first pipe character (|).

Usage *[form!]vsElastic*.**TagSplit** [= {**True**|**False**}]

Remarks This property is useful if you want to use the Elastics tag labels feature but you need to store private information on the Tags as well. For example, if you want to store some help information on the Tag strings in addition to the labels, you could use the following code:

```
Sub Form_Load ()
   vsElastic.TagSplit = True
   vsElastic.TagWidth = -100
   Ctrl(0).Tag = Label 0|Help for control 0
   Ctrl(1).Tag = Label 1|Help for control 1
   Ctrl(2).Tag = Label 2|Help for control 2
End Sub

Sub Ctrl_GotFocus (Index As Integer)
   HelpLabel = Mid$(Ctrl(Index).Tag, Instr$(Ctrl(Index).Tag, |))
End Sub
```

This property only applies if the TagWidth property is set to a non-zero value.

Default Value False

Data Type Boolean

TagWidth Property [3]

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 can be placed on the left, right, above or below the control being labeled, depending on the setting of the TagPosition property.

When TagPosition is set to *0 - Left* or *3 - Right,* the labels have the same height as the controls being labeled. The width of the labels is controlled by the TagWidth property, as explained below.

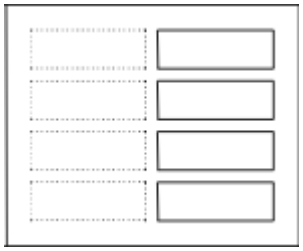
When TagPosition is set to *1 - Above* or *3 - Below,* the labels have the same width as the controls being labeled. The height of the labels is controlled by the TagWidth property, as explained below.

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.

The picture below illustrates the effect of different TagWidth and TagPosition settings. Dotted rectangles indicate tag labels, solid rectangles indicate controls being labeled. All controls are assumed to be 1000 Twips wide and 100 Twips tall.

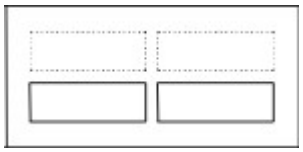


The following code resizes the labels so theyre always as **wide** as the controls:

```
vsElastTagPosition = 0 ' left
vsElastTagWidth = -100 ' 100%
```

The following code keeps the labels 1000 Twips **wide**:

```
vsElast.TagPosition = 0 ' left
vsElast.TagWidth = 1000 ' 1000 twips
```



The following code resizes the labels so theyre always as **tall** as the controls:

```
vsElastTagPosition = 1 ' above
vsElastTagWidth = -100 ' 100%
```

The following code keeps the labels 100 Twips **tall**:

```
vsElastTagPosition = 1 ' above
vsElast.TagWidth = 100 ' 100 twips
```

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. The distance between the tag labels and the controls is determined by the ChildSpacing property.

If you change a controls Tag property at run time, you must tell the Elastic to update the tag label. You can accomplish this with the Refresh method, as in the example below:

```
' Assuming Picture1 is inside VsElastic1
Picture1.Tag = New Label
VsElastic1.Refresh
```

Default Value Zero

Data Type Long

Template Property [5]

Description This property allows you to set a number of other properties at once, in order to achieve a pre-defined type of Elastic.

Usage *[form!]vsElastic*.**Template** [= *setting%*]

Remarks Valid settings for this property are:

0 - None
1 - Data Entry
2 - Tool Bar
3 - Status Line
4 - Picture Button
5 - Captioned Button

The table below summarizes the effect of each setting:

Setting	**Effect**	
0 - None	None	
1 - Data Entry	AccessKey = True	AutoSizeChildren = EVENVERT
	FontBold = False	TagWidth = -100
2 - Tool Bar	Align = TOP	AutoSizeChildren = EVENHORZ
	BevelOuter = RAISED	BevelOuterDir = HORZ
	BevelOuterWidth = 1	BevelInner = NONE
	BorderWidth = 2	ChildSpacing = 1
	MaxChildSize = 400	MinChildSize = 400
3 - Status Line	Align = BOTTOM	AutoSizeChildren = EVENHORZ
	BevelOuter = RAISED	BevelOuterDir = HORZ
	BevelOuterWidth = 1	BevelInner = NONE
	BorderWidth = 2	ChildSpacing = 1
	FontBold = FALSE	
	MaxChildSize = 400	MinChildSize = 400
4 - Picture Button	AccessKey = False	
	BevelInner = NONE	BorderWidth = 2
	CornerColor = LTGRAY	FontBold = False
	Style = COMMAND	
	ShowFocusRect = FALSE	ShowOutline = FALSE
	PicturePos = CNTRCNTR	TabStop = FALSE
5 - Captioned Button	AccessKey = True	
	BevelInner = NONE	BorderWidth = 2
	CaptionPos = CNTRBTM	
	CornerColor = LTGRAY	FontBold = False
	Style = COMMAND	
	ShowFocusRect = TRUE	ShowOutline = TRUE
	PicturePos = CNTRTOP	TabStop = TRUE

Default Value 0 - None

Data Type Integer

Version Property [5]

Description This property returns the version of the VSVBX controls currently loaded in memory.

Usage *CheckVer% = [form!]vsElastic*.**Version**

Remarks You may want to check this value at the Form_Load event, to make sure the version that is executing is at least as current as the version used to develop your application.

The version number is a three digit integer where the first digit represents the major version number and the last two represent the minor version number. For example, version 5.00 would return 500.

This property is read-only.

Data Type Integer

WordWrap Property [3]

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

Data 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 VSOCX16.OCX, VSOCX32.OCX or VSVBX.VBX

Object Type vsIndexTab

Note Before you can use a vsIndexTab control in your application, you must add VSOCX/VSVBX to your project (see the Visual Basic manual for details). To automatically include VSVBX/VSOCX in new projects, put it in an AUTOLOAD file. When distributing your application, you should install the VSVBX/VSOCX files in the user's Microsoft Windows SYSTEM subdirectory.

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	* CaptionStyle [5]	* ClientLeft
* ClientHeight	* ClientTop	* ClientWidth
* CurrTab	* DogEars [4]	DragIcon
DragMode	Enabled	* FirstTab
FontBold	FontItalic	FontName
FontStrikethru	FontUnderline	ForeColor
* FrontTabColor	* FrontTabForeColor	* GetTab [5]
Height	HelpContextID [3]	Hwnd [3]
Index	Left	* MouseOver [4]
MousePointer	* MultiRow [4]	* MultiRowOffset [5]
Name	* New3D	* NumTabs
* Position	* SetParent [5]	* ShowFocusRect [3]
* Style	* TabCaption [3]	* TabColor [4]
* TabEnabled [4]	* TabHeight [5]	TabIndex
* TabOutlineColor [3]	* TabPicture [4]	* TabPreview [5]
* TabsPerPage [3]	TabStop	Tag
* Template [5]	Top	Visible
* Version [5]		

Events

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

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

Data 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. To activate a tab, double-click it with the mouse. 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, double-click the current tab with the right mouse button. This causes all tabs to be de-selected and the containers to be tiled in tab order. Reorganize the tabs by dragging them around until they are in the proper order.

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

Data Type Boolean

BackSheets Property [3]

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 MultiRow is set to False.

Default Value 2 - Sheets

Data 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
Data Type	Long (color)

BoldCurrent Property [4]

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

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

Default Value True

Data Type Boolean

BorderWidth Property [4]

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

Data 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.

Data Type String

CaptionStyle Property [5]

Description Allows you to specify a 3D effect for the IndexTabs caption.

Usage *[form!]vsIndexTab*.**CaptionStyle** [= *setting%*]

Remarks Valid settings for this property are:

0 - Flat
1 - Raised
2 - Inset
3 - Raised Light
4 - Inset Light

Settings 1 and 2 work best for large and bold fonts. Settings 3 and 4 work best for small regular fonts.

Default Value 0 - Flat

Data Type Integer

Click Event

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

Syntax **Sub** *vsIndexTab*_**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.

Data 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

Data Type Integer

DogEars Property [4]

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

Data 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

Data 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

Data Type Long (color)

FrontTabForeColor Property [4]

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

Data Type Long (color)

GetTab Property [5]

Description This property allows you to determine the tab page on which a given control is.

Usage *TabNumber% = [form!]vsIndexTab*.**GetTab(***[form!]Control*.**hWnd)**

Remarks If you pass GetTab an invalid window handle, or a window that does not belong to any tab, GetTab will return the value -1.

This property is read-only.

Data Type Integer

MouseOver Property [4]

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.

Data Type Integer

MultiRow Property [4]

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 become visible at once.

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

Data Type Boolean

MultiRowOffset Property [5]

Description This property allows you determine the offset, in Twips, between successive tab pages.

Usage *[form!]vsIndexTab*.**MultiRowOffset** [= *setting&*]

Remarks This property only works if MultiRow is set to True.

Default Value 200

Data Type Long (Twips)

New3D Property

Description Determines whether vsIndexTab should use traditional or contemporary 3-D effects when drawing the tabs.

Usage *[form!]vsIndexTab*.**New3D** [= {**True**|**False**}]

Remarks This property only has an effect when the Style property is set to one of the 3-D styles.

Default Value False

Data 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.

Data 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

Data Type Integer

SetParent Property [5]

Description Allows you to add pages to an IndexTab at run time.

Usage *[form!]vsIndexTab*.**SetParent** = *NewPage.Hwnd*

Remarks After adding pages to an IndexTab at run-time, you must sort them in the correct order by setting each pages Left property to the Left property of the previous page plus an offset. The code below shows how this is done:

```
Sub Form_Load ()
  Dim I%

  ' add pages to vsindextab
  For i% = 0 To 2
    vsindextab1.SetParent = picture1(i).hWnd
  Next

  ' sort pages into appropriate order
  For i% = 1 To 2
    picture1(i).Left = picture1(i - 1).Left + 100
  Next
End Sub
```

Setting this property to an invalid Hwnd value will trigger a run-time error. Invalid Hwnd values are those that do not correspond to a window, those of windows that contain the Elastic, and those of top-level windows (forms).

You cannot remove pages using this property. If you need to do that, declare and use the Windows API SetParent function.

This property is write-only.

Data Type Integer (Hwnd)

Scroll Event

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

Syntax **Sub** *vsIndexTab*_**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

Data 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]
7 - Straight plus 3D effect [4]
8 - CutCorners [5]
9 - CutCorners 3D [5]

Default Value 0 - Slanted Lines

Data Type Integer

Switch Event [5]

Description Fired immediately before the current tab is changed.

Syntax **Sub** *vsIndexTab*_**Switch (OldTab As Integer, NewTab As Integer, Cancel As Integer)**

Remarks This event is useful when you have to validate data or save the contents of controls on the current tab page before switching active tabs.

The differences between the Switch and the Click events are:

Switch only fires when the current tab is actually going to be changed. Click also fires when you click the current tab or the client area.

Switch passes the old and new tabs as parameters, and it allows you to cancel the switch by setting the Cancel parameter to True.

TabCaption Property [3]

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
```

Data Type String array

TabColor Property [4]

Description Sets or retrieves the color for an individual tab.

Usage *[form!]vsIndexTab*.**TabColor(***index%***)** [= *color&*]

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.

Data Type Long Array (color)

TabEnabled Property [4]

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.

Data Type Boolean array

TabHeight Property [5]

Description This property allows you to specify the height, in Twips, of each tab.

Usage *[form!]vsIndexTab*.**TabHeight** [= *height&*]

Remarks If you set this property to zero, the IndexTab determines the height of each tab automatically, based on the current font and size of the tab pictures.

DefauLt Value 0

Data Type Long (Twips)

TabOutlineColor Property [3]

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

Data Type Long (color)

TabPicture Property [4]

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

Usage *[form!]vsIndexTab*.**TabPicture(***index%***)** [= *picture%*]

Remarks You can assign bitmaps, icons or metafiles to tabs. 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.

Data Type Picture array

TabPreview Property [5]

Description Determines whether tab pages should be shown when the user clicks and drags over the tabs.

Usage *[form!]vsIndexTab*.**TabPreview** [= {**True**|**False**}]

Remarks This property only works for single-row tabs. Multi-row tabs always behave as if TabPreview were set to False.

Default Value True

Data Type Boolean

TabsPerPage Property [3]

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

Data Type Integer

Template Property [5]

Description This property allows you to set a number of other properties at once, in order to achieve a pre-defined type of IndexTab.

Usage *[form!]vsIndexTab*.**Template** [= *setting%*]

Remarks Valid settings for this property are:

0 - None
1 - Redmond
2 - Scotts Valley
3 - Cambridge
4 - Berkeley
5 - Chicago

Look up the Template property in your VSVBX manual for a picture of what each setting looks like. When designing your tabs, choose the template closest to the look you want, then customize it further by setting other properties.

This property is write-only. After you select a template, the value shown in the Properties window will automatically revert to *0 - None.*

Default Value 0 - None

Data Type Integer

Version Property [5]

Description This property returns the version of the VSVBX controls currently loaded in memory.

Usage *CheckVer% = [form!]vsIndexTab*.**Version**

Remarks You may want to check this value at the Form_Load event, to make sure the version that is executing is at least as current as the version used to develop your application.

The version number is a three digit integer where the first digit represents the major version number and the last two represent the minor version number. For example, version 5.00 would return 500.

This property is read-only.

Data 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 VSOCX16.OCX, VSOCX32.OCX or VSVBX.VBX

Object Type vsAwk

Note Before you can use a vsAwk control in your application, you must add VSOCX/VSVBX to your project (see the Visual Basic manual for details). To automatically include VSVBX/VSOCX in new projects, put it in an AUTOLOAD file. When distributing your application, you should install the VSVBX/VSOCX files in the user's Microsoft Windows SYSTEM subdirectory.

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]	* FindLine [5]	* FS
Index	* L	* MatchQuotes [3]
Name	* NF	* PercentDone
* PosAt [3]	* RN	Tag
* Val [4]	* Version [5]	

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.

Data Type Integer

Begin Event

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

Syntax **Sub** *vsAwk*_**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

Data 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.

Before you get or set the CurrPos property, you must open the file by setting the Action property to *0 - Scan File* or *1 - Get Line*.

After you set the CurrPos property, the file handle will be repositioned. You must then read the line by setting the Action property to *1 - Get Line*.

The example below illustrates how you can do this:

```
' look through a file for for a line containing a string,
' return the position where it was found or -1 if not found
Function GetStringPosition (fname$, fstr$) As Long
   vsawk1.FileName = fname              ' set file name
   vsawk1.FindLine = fstr               ' look for string
   vsawk1.Action = 2                    ' close file
   If vsawk1 = "" Then                  ' not found
      GetStringPosition = -1
      Debug.Print fstr; " Not found in "; fname
   Else                                 ' found
      GetStringPosition = vsawk1.CurrPos
   End If
End Function

' read a line at a specified position in a file
Sub ReadAtPosition (savepos&)
   vsawk1.Action = 1                    ' force file open
   vsawk1.CurrPos = savepos             ' go to position
   vsawk1.Action = 1                    ' read line
   Debug.Print vsawk1                   ' show line
   vsawk1.Action = 2                    ' close file
End Sub
```

Data Type Long

End Event

Description Fired when Awk is done scanning a file.

Syntax **Sub** *vsAwk*_**End ()**

Error Event

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

Syntax **Sub** *vsAwk*_**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: These error may occur while parsing a file.

Code	Name	Description
0	*No Error*	No error detected
1	*Can't open file*	The file specified by the FileName property can't be found.
2	*Line too long*	The file contains lines longer than 64k characters. In this case, Awk truncates the line.
3	*Field too long*	Changing the current feld would make the current line longer than 64k characters.
4	*Bad field index set*	Setting a field that doesn't exist, i.e., smaller than zero or greater than NF.
5	*Bad field index get*	Getting a field that doesn't exist, i.e., smaller than zero or greater than NF.

Expression evaluator errors: These error may occur while evaluating an expression using the Val property.

Code	Name	Description
6	*Syntax*	Expression has bad syntax, such as **1+*2**.
7	*Bad token*	Tried to evaluate an expression with undefined variables, such as **a+b**.
8	*Unbalanced brackets*	Expression has unbalanced brackets, such as **1+(2+3.**
9	*Division by zero*	Tried to divide an expression by zero.
10	*Domain*	Argument is not in domain of function, such as **Log(-1)**.
11	*Singularity*	Expression would result in a singularity.
12	*Overflow*	Expression would produce a result too big to represent, such as **Exp(1000).**
13	*Partial loss of significance*	Expression would produce a result with partial loss of significant digits.
14	*Total loss of significance*	Expression would produce a result with total loss of significant digits, such as **Sin(10^70)**.
15	*Underflow*	Expression would produce a result too small to represent.
16	*Not reentrant*	Tried to use the Val property while handling the Variable event.

Data 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
```

Data 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
```

Data 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.

Data Type String.

FileType Property

Description Determines whether the file being scanned is a text or binary file.

When FileType is set to *0 - Text File*, the Awk recognizes line breaks whether they are represented by CR/LF [chr$(13)+chr$(10)] combinations or single LF characters [chr$(10)], and trims both off the end of each line. In addition, the Awk will truncate the file when it finds an end-of-file marker (ctrl-Z, chr$(6)).

When FileType is set to *1 - Binary File*, the Awk will perform no conversions whatsoever on the file being scanned.

Usage *[form!]vsAwk*.**FileType** [= *setting%*]

Remarks Valid settings for this property are:

0 - Text File
1 - Binary File

Default Value 0 - Text File.

Data Type Integer

FilterQuotes Property [4]

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.

Data Type Boolean.

FindLine Property [5]

Description Allows you to scan a file for a line containing a string.

Usage *[form!]vsAwk*.**FindLine** = *searchstring$*

Remarks Before using this property, you must set the FileName property to the name of the file you want to search.

After setting this property, the line containing the string will be returned in the L (line) property and the CurrPos property will be set to the position within the file where the line was found.

If Awk cannot find the line containing the given string, the L (line) property is set to an empty string.

The search is case-sensitive. If you want to ignore the case, set the Case property to *1 - Upper* and use an upper case search string.

For an example, see the documentation for the CurrPos property.

This property is write-only.

Data Type String.

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)

Data 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
```

Data Type String.

MatchQuotes Property [3]

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

Data 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.

Data 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.

Data Type Integer

PosAt Property [3]

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
```

Data Type Integer array.

RN (Record Number) Property

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

Usage *[form!]vsAwk*.**RN**

Data 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** *vsAwk*_**Scan ()**

Val Property [4]

Description Evaluates the mathematical expression in the Line 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, Log10, Exp, Sqr, Int, Frac, Ceil,** and **Floor**. Trignonmetric functions accept and return angles expressed in radians.

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 an appropriate code. See the description of the Error property to find out what each code means.

Example:

```
VsAwk1 = "(1+2)*3^2"
debug.print VsAwk1.Val
  27
```

Data Type Single for VBX,
Double for OCX
(If you need double-precision results with the VBX version, see **Double-Precision** in the Hints and TroubleShooting Section.)

Variable Event [4]

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

OCX::
Sub *vsAwk*_**Variable (***Variable As String, Value As Double, Accept As Integer***)**
VBX::
Sub *vsAwk*_**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:

```
'
' OCX version
'
Sub VsAwk1_Variable (Variable$, Value As Double, Accept%)
  Accept = True
  Select Case Variable
    Case "X", "x": Value = GlobalX
    Case "Y", "y": Value = GlobalY
    Case Else:     Accept = False ' trigger error
  End Select
End Sub
'
' VBX version
'
Sub VsAwk1_Variable (Variable$, Value As Single, Accept%)
  Accept = True
  Select Case Variable
    Case "X", "x": Value = GlobalX
    Case "Y", "y": Value = GlobalY
    Case Else:     Accept = False ' trigger error
  End Select
End Sub
```

Version Property [5]

Description This property returns the version of the VSVBX controls currently loaded in memory.

Usage *CheckVer% = [form!]vsAwk*.**Version**

Remarks You may want to check this value at the Form_Load event, to make sure the version that is executing is at least as current as the version used to develop your application.

The version number is a three digit integer where the first digit represents the major version number and the last two represent the minor version number. For example, version 5.00 would return 500.

This property is read-only.

Data Type Integer

Hints and 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.

Invisible controls

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

Explanation When the AutoSize property is set to something other than *7 Proportional*, the Elastic ignores invisible controls (there would be gaps between the visible controls if it didnt). 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
```

Flickering Elastics

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 - Use the QuickPaint property to force instant redrawing. This consumes some memory, and there may be a little delay while the Elastic builds its new image in memory. But it eliminates flicker.

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

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

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

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

Disabled controls

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.

Uneven multi-rows

Trouble I have a multi-row IndexTab with an uneven number of tabs (that is, the last row is incomplete). When I switch tabs, the rows get mixed.

Explanation The IndexTab always tries to populate the lowest rows, even if it has to change which tabs go into each row.

Solution Make the number of tabs even by adding invisible tabs. Invisible tabs are those with a caption that starts with a dash character.

For example, if you have TabsPerPage set to 4, and you only need 6 tabs, add two invisible tabs so the total becomes 8, a multiple of 4.

```
' ** Before
vsIndexTab1.Caption = 1|2|3|4|5|6

' ** After
vsIndexTab1.Caption = 1|2|3|4|5|6|-7|-8
```

VideoSoft Products

To get an order form, click **HERE**.

VSOCX/VSVBX

A set of three custom controls for interface design and text parsing.

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	Allows 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.

VSVIEW

A set of four custom controls for creating, viewing, and printing text and graphics.

Icon	Name	Object	Description
	InForm	vsInForm	A control that you can drop into any container to customize its title bar, frame, resizing behavior, and frame buttons. InForm also allows you to monitor the clipboard, drag and drop files from File manager, and more.
	Printer	vsPrinter	A much improved printer object with word wrap, headers and footers, multi-column printing, graphics, and multi-page Print Preview capability.
	ViewPort	vsViewPort	A control that gives you a scrollable virtual area so you can fit more controls in your windows. Great for implementing Print Preview and programs that look like the Program Manager.
	Draw	vsDraw	A versatile drawing control that lets you create complex images, view them on the screen, copy them to the clipboard, or print them. Great for technical drawings, maps, and diagrams.

VSFLEX

A set of two custom controls for analyzing, formatting, and displaying information.

Icon	Name	Object	Description
	FlexArray	vsFlexArray	A new way to display and operate on tabular data. FlexArray gives you total flexibility to display, sort, merge, and format tables containing strings and pictures.
	FlexString	vsFlexString	A powerful regular expression engine. With FlexString, you can find and replace patterns in strings. Use it to provide regular expression search-and-replace capabilities or to parse input strings.

Order Form

(You may print this form by selecting the File|Print command).

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

To order by phone, call
(800) 547-7295 (from within the US)
(510) 704-8200 (from anywhere)
(510) 843-0174 (fax)

Please register my copy of the following VideoSoft products. I am enclosing a check or money order for the amount of:

OCX Version (includes VBX)		
VSOCX.OCX	Single developer	US$ 99.00
	Additional developers	__ x 99.00
VSVIEW.OCX	Single developer	149.00
	Additional developers	__ x 149.00
VSFLEX.OCX	Single developer	149.00
	Additional developers	__ x 149.00
VBX Version		
VSVBX.VBX	Single developer	US$ 45.00
	Additional developers	__ x 45.00
VSVIEW.VBX	Single developer	99.00
	Additional developers	__ x 99.00
VSFLEX.VBX	Single developer	99.00
	Additional developers	__ x 99.00

Note: Call us for details on site licenses and volume discounts.

Name:

Company:

Street:

City, State, ZIP:

Country:

Phone:

Where did you hear about the VideoSoft products?