Templar List Collection v2.0

Ctree

EnhList

ImageBag

StyleBag

TSLabel

TabRas

TSTree

MTree

LParse

The Templar List Collection is a powerful set of list and tree controls. They are used to manipulate, query, and display user data. The List Collection is currently comprised of nine controls. There are seven displayable controls (CTree, EnhList, ImageBag, StyleBag, TabRas, TSTree, and TSLabel) and two hidden controls (MTree and LParse).

Controls

Properties

Methods

Events

Codes and Constants

About Templar Software

Standard Programming Interface

All of the List Collection Controls share the same set of standard list-interface properties (ListCount, ListIndex, NewIndex, List, ItemData). This allows the programmer to become familiar with the Templar List Controls fairly easily, and provide a standard way to manipulate data from within each of the controls.

Memory and Speed

List Collection Controls also provides many features not found in other controls. Primarily, they all use a Virtual Memory subsystem which allow them to potentially access up to 500 Megabytes of RAM. This memory can be used to store lots data. In fact, each item in a list can store over 128K of data! The BatchMode property also helps to speed up intensive operations by postponing redundant calculations and paints until the end of the operation. This is great when you're adding a lot of items at once.

Borders and Shadows

The Border properties allow you to display ThreeD Borders and Shadows. You can determine the dimensions of your border by using the BorderWidth, BevelOuter, BevelInner, and ShadowWidth properties. Customize the look of your border with the BevelLight, BevelDark, ShadowWidth and

TrueShadow properties.

Images and Styles

By using the ImageBag control, an almost unlimited number images (constrained only by available memory, of course) of varying dimensions and scales can be used to provide displayable images to other controls in the List Collection. This helps to conserve memory for often used images and also lets you create standardized images for use throughout your application. On the other hand, you can keep several ImageBags handy and change the appearance of your app on the fly.

Like the ImageBag, the StyleBag provides a convenient repository for often used item styles. Item styles can be used to greatly improve the look and feel of your program. Each item style not only lets you set all of the font and color attributes for an item, but also let you display Three-D borders and shadows around items. Again, you can mix and match StyleBags with other List Collection controls.

Display and Special Effects

Many of the controls also support Control Flags. By using the ControlFlags property, you can control the display of the control's border, scrollbars, and focus rectangle. Certain special effects such as auto-expand/collapse, automatic scrolling, multiple selection, and item layout can also be set with this property.

Akin to Control Flags are Item Flags. These are flags which you can set for each item to define that item's image placement, text wrapping, text threed and shadow effects, and highlighting. When connected to a StyleBag control, each item also has the ability to display borders, bevelling, and shadows, also controlled by the ItemFlags( ) and StyleFlags( ) (for the StyleBag properties.)

Formatted Display

The List Collection controls stand apart from many other types of listboxes by offering properties to display images, multi-colored text, and indention. Displayed text can be formatted by embedding formatting commands into the List( ) Property. Formatting commands are available to change font attributes, colors, and text placement. Since a format code can be placed anywhere in an item's text, you can display text with different font attributes and colors within a single item!

Hierarchies and Trees

Some of the List Collection controls can be used to display data in an outline or hierarchical format. The CTree and TSTree controls let you set the indentation level of different items to establish parent-child relationships in your list that can be accessed by using the control's Tree properties. The MTree (an invisible memory-only control) and the TSTree both store hierarchical data in a somewhat different fashion. These controls provide you with special properties to navigate through lists of items, wherein each item in the list may have a sub-list of its own; a list of lists.

Data Storage

Each item in your list comes with a wide-range of properties which make them great, not only for data display but data storage as well. Each Item has a List (or ItemText) property to display formatted text in the display area of the control. Additional properties include ItemData and ItemString (also called ExtraString), used to store non-visible data associated with an item. An Item also has properties for item size, indentation, formatting, visibility, and disabling.

Searching

Another set of valuable properties found in these controls are the StartSearch, SearchString, FoundIndex, and SearchType properties. By using these properties, locating data within the control becomes an automated process and, consequently, more efficient than looping through and checking each item yourself.

Enhanced Events

In addition to many of the standard events available, there are several custom events that help you to write efficient and structured code.   The IndexChange, ItemClick, ItemDblClick, and ItemOver events all provide you with information regarding the current selection, button presses, key presses, and mouse location.

Head in the Clouds ...

The Templars were an ancient order of warrior-monks chartered to provide protection to pilgrims travelling to the holy lands of the time. Renowned for their ferocity in combat and dedication to the spirit of truth and honesty, they are currently the source for many of our chivalric legends and speculations. Today, Templar Software participates in the same tradition of taking our charge seriously and with great respect. Our charge, of course, is empowering programmers with great software tools and applications. Have fun.

... Feet on the Ground

Templar Software is really just a bunch of hermit-types who have fallen in love with programming and the ongoing Information Revolution. The thought of what the future might hold tends to thrill and inspire us. Like good little warriors, we are always looking for new challenges and ways to hone our skills.

Contracting with Templar Software :

Hey! We're not just your favorite component guys, we write apps too! That's right, Templar Software is also available for contracts and custom work. Over the years (since 1983), we have been working with organizations both large and small in order to improve their efficiency and allow them to work in an environment where educated decisions are fostered through an increased amount of available information.

When designing an interface, we labor to make it accessible to varying degrees of users. Beginners are able to work without being overwhelmed by too many options. The main functionality of the program is brought to the forefront where most of the time is spent. We have found this to be a crucial factor in the acceptance of any new software. As users become more accustomed to the workings of the program, they find settings which allow them to customize the interface or let them view their data in a different way. This helps experienced users focus on the details that are important to them.

So, whether it's writing an in-office app or system, consultation on product design, or the customization of a Templar product, please give us a call and let us know if we can be of help. We also give special consideration (i.e. charge less) for especially challenging tasks!

Contacting Templar Software :

Templar Software
1125 Duke Street
Alexandria, VA 22314

Voice : (703) 518-4480
Fax : (703) 519-0030
Compuserve : 72400, 2345
Internet : ABowlin@nmaa.org

Getting Information on Templar Products :

We are frequently upgrading and adding to the Templar List Collection. Please check the following area for updates and information regarding any of our products. Registered users will enjoy many benefits such as free upgrades, participation in our beta program, can contribute tips, tricks, demos, etc.

http://www.vbxtras:80.com/~Templar

Purchasing Templar Software Products :

Templar Software currently processes all sales orders through VBxtras, Inc. This is cool because not only do they give you a nice little discount off of our $140 retail price, but you'll also get a copy of their rather extensive catalog. Tell them Templar Software sent you (maybe we'll finally win a popularity contest or something). Anyway, here's their info :

VBxtras, Inc.
1905 Powers Ferry Road
Suite 100
Atlanta, GA 30339

Voice : (800) 788-4794 or (404) 952-6356
Fax : (404) 952-6388

Codes and Constants Overview

See also :

Templar List Collection v2.0

In the Templar List Collection, the controls (and their items, if applicable) use a variety of flags and codes for cutomizing the display.

Control Flags
Item Flags
Text Formatting Codes
Constants

Control Flags

See also :

ControlFlags Property
Codes and Constants Overview

Note: Using flags to set a number of display options is similar to the way the MS Visual Basic Message Box(MsgBox) flags work. This allows you to specify many display options in one line of code. When you specify flags for both the Item or the Control you may use either the string value or its numerical equivalent.

Applies To :

ImageBag
StyleBag
TabRas
TSLabel
TSTree

Control Border Flags:

String Name	Value	Description
BRD_SHADOW	1	Border Shadow for Control
BRD_PUSHED	2	Control Border looks Pushed when set (use with BRD_SHADOW)
BRD_BOUTLINE	4	Draw an Outline on the Outside edge of the border
BRD_BORDER	8	Display the Border
BRD_BEVEL	16	Display a Bevelled Border
BRD_COUTLINE	32	Draw an Outline on the Inside edge of the border
BRD_ALL	-	(BRD_SHADOW+BRD_OUTLINE+BRD_BORDER+BRD_BEVEL+BRD_COUTLINE)

Scroll Flags:

String Name	Value	Description
VSC_SHOW	64	Show the Vertical Scrollbar
VSC_CLICKBORDER	128	Scroll Vertically by Clicking on the Top or Bottom Borders
VSC_OVERBORDER	256	Scroll Vertically by Moving the Mouse Over the Top or Bottom Borders
VSC_ALL	-	(VSC_SHOW+VSC_CLICKBORDER+VSC_OVERBORDER)
HSC_SHOW	512	Show the Horizontal Scrollbar
HSC_CLICKBORDER	1,024	Scroll Horizontally by Clicking on the Left or Right Borders
HSC_OVERBORDER	2,048	Scroll Horizontally by Moving the Mouse Over the Left or Right Borders
HSC_ALL	-	(HSC_SHOW+HSC_CLICKBORDER+HSC_OVERBORDER)

Focus Rectangle Flags:

String Name	Value	Description
FOC_TEXT	4,096	Draw Focus Rect around Text of Current Item
FOC_ITEM	8,192	Draw Focus Rect around All of Current Item
FOC_CONTROL	16,384	Draw Focus Rect around the Inside of the Display area of the Control
FOC_ALL	-	(FOC_TEXT+FOC_ITEM+FOC_CONTROL)

Selection Flags:

String Name	Value	Description
SEL_LCLICK	32,768	Select the Current Item with a Left Mouse Click

SEL_RCLICK	65,536	Select the Current Item with a Right Mouse Click
SEL_OVER	131,072	Select the Current Item by Moving the Mouse Over it
SEL_MULT	262,144	Allow Multiple Selection (use Shift and Ctrl Keys while Selecting
SEL_ALL	-	(SEL_LCLICK+SEL_RCLICK+SEL_OVER+SEL_MULT+SEL_HILITE)

Expand/Collapse Flags: (TabRas and TSTree only)

String Name	Value	Description
EXP_LCLICK	524,288	Expand/Collapse Item with a Left Mouse Click
EXP_RCLICK	1,048,576	Expand/Collapse Item with a Right Mouse Click
EXP_LDBLCLICK	2,097,152	Expand/Collapse Item with a Left Mouse Double-Click
EXP_RDBLCLICK	4,194,304	Expand/Collapse Item with a Right Mouse Double-Click
EXP_ALL	-	(EXP_LCLICK+EXP_RCLICK+EXP_LDBLCLICK+EXP_RDBLCLICK)

Layout Flags: (TabRas only)

String Name	Value	Description
LAY_VERT	8,388,608	Arrange Items Vertically, from Top to Bottom
LAY_HORZ	16,777,216	Arrange Items Horizontally, from Left to Right
LAY_WRAP	33,554,432	Wrap Items as they exceed the Bottom or Right edge of the Control
LAY_ALL	-	(LAY_VERT+LAY_HORZ+LAY_WRAP)

Miscellaneous Flags:

String Name	Value	Description
CUR_HILITE	67,108,864	Always Highlight the Current Item

Item Flags

See also :

ItemFlags( ) Property
Codes and Constants Overview

Note: Using flags to set a number of display options is similar to the way the MS Visual Basic Message Box(MsgBox) flags work. This allows you to specify many display options in one line of code. When you specify flags for both the Item or the Control you may use either the string value or its numerical equivalent.

Applies To :

TabRas
TSLabel
TSTree

Item Border Flags:

String Name	Value	Description
BRD_SHADOW	1	Border Shadow for Control
BRD_PUSHED	2	Control Border looks Pushed when set (use with BRD_SHADOW)
BRD_BOUTLINE	4	Draw an Outline on the Outside edge of the border
BRD_BORDER	8	Display the Border
BRD_BEVEL	16	Display a Bevelled Border
BRD_COUTLINE	32	Draw an Outline on the Inside edge of the border
BRD_ALL	-	(BRD_SHADOW+BRD_OUTLINE+BRD_BORDER+BRD_BEVEL+BRD_COUTLINE)

Image Flags:

String Name	Value	Description
IMG_TOP	64	Image Above Text
IMG_LEFT	128	Image to Left of Text
IMG_RIGHT	256	Image to Right of Text
IMG_BOTTOM	512	Image Below Text
IMG_ALL	-	Center Image (use with TXT_TRANS Flag)

Text Flags:

String Name	Value	Description
TXT_WRAP	1,024	Wrap Text to Fit in Width (use with ItemWidth Property)
TXT_LEFT	2,048	Align Text to Left
TXT_RIGHT	4,096	Align Text to Right
TXT_CENTER	8,192	Align Text in Center
TXT_THREED	16,384	Draw Three-D Text (use with TXT_SHADOW to heighten effect)
TXT_SHADOW	32,768	Draw Shadowed Text
TXT_TRANS	64,536	Draw Text in Transparent mode (used with IMG_ALL Flag)
TXT_ALL	-	

Highlight Flags:

String Name	Value	Description
HIL_TEXT	131,072	Highlight Text Only
HIL_ITEM	262,144	Highlight whole Item (Text and image)
HIL_ALL	-	(HIL_TEXT + HIL_ITEM)

Text Formatting Codes

See also :

Codes and Constants Overview
Example

Note: Using Text Formatting Codes allows you to embed different font colors, bolding, etc. into the middle of the ItemText (List( )) being displayed to the screen.

Applies To :

CTree
TabRas
TSLabel
TSTree

Font Attributes

- \B - Bold On
- \b - Bold Off
- \I - Italics On
- \i - Italics Off
- \U - Underline On
- \u - Underline Off

- **\S - StrikeThru On**
- **\s - StrikeThru Off**

Color

- **\CFn\ or \CFccc\ - Text Foreground**
- **\CBn\ or \CBccc\ - Text Background**
- **\CRn\ or \CRccc\ - Fill Rectangle Color**

Where n and ccc are :

0	BLK	-	Black
1	BLU	-	Blue
2	GRN	-	Green
3	CYN	-	Cyan
4	RED	-	Red
5	MAG	-	Mag
6	YEL	-	Yellow
7	WHT	-	White
8	GRY	-	Gray
9	LTB	-	Light Blue
10	LTG	-	Light Green
11	LTC	-	Light Cyan
12	LTR	-	Light Red
13	LTM	-	Light Magenta
14	LTY	-	Light Yellow
15	LTW	-	Light White

- **\CD - Restore Default Colors**

Horizontal Placement

- **\TCn\ - Set New Output Location to n Characters from the Left**
- **\TXn\ - Set New Output Location to n Pixels from the Left**

Constants

Note: All constants are located in a file called **TLC20.BAS**.

```
' DisplayType Constants (CTREE)
Global Const DISPLAYTYPE_REGULAR = 0
Global Const DISPLAYTYPE_FORMATTED = 1
Global Const DISPLAYTYPE_RAWTEXT = 2

' DrawMethod Constants (ENHLIST & CTREE)
Global Const DRAWMETHOD_NORMAL = 0
Global Const DRAWMETHOD_SMOOTH = 1

' FileAction Constants (ENHLIST)
Global Const FILEACTION_NONE = 0
Global Const FILEACTION_READ = 1
Global Const FILEACTION_WRITE = 2
Global Const FILEACTION_DELETE = 3

' HighlightStyle Constants (ENHLIST & CTREE)
Global Const HIGHLIGHTSTYLE_NONE = 0
Global Const HIGHLIGHTSTYLE_TEXT = 1
Global Const HIGHLIGHTSTYLE_BAR = 2

' ListType Constants (LPARSE)
Global Const LISTTYPE_PARSEDSTRING = 0
Global Const LISTTYPE_KEYWORDS = 1
Global Const LISTTYPE_SYMBOLS = 2

' MoveCurr Constants (MTREE)
Global Const MOVECURR_NONE = 0
Global Const MOVECURR_FIRST = 1
Global Const MOVECURR_LAST = 2
Global Const MOVECURR_PREV = 3
Global Const MOVECURR_NEXT = 4
Global Const MOVECURR_PARENT = 5
Global Const MOVECURR_ROOT = 6

' SearchType Constants (ALL Controls)
Global Const SEARCHTYPE_LIST = 0
Global Const SEARCHTYPE_ITEMDATA = 1
Global Const SEARCHTYPE_EXTRASTRING = 2

' TokenType Constants (LPARSE)
Global Const TOKENTYPE_NONE = -1
Global Const TOKENTYPE_IDENTIFIER = 0
Global Const TOKENTYPE_KEYWORD = 1
Global Const TOKENTYPE_SYMBOL = 2
Global Const TOKENTYPE_WHITESPACE = 3
Global Const TOKENTYPE_QUOTEDSTRING = 4
Global Const TOKENTYPE_NUMERIC = 5

'==============================================
'===                                        ===
'===          Additions for v2.0            ===
'===                                        ===
'===============================================

' Control Messages
Global Const CCMSG_BASE = (&H7F00& + &HAB&)
Global Const CCMSG_SERIALIZEPROPS = (CCMSG_BASE + 0)
Global Const CCMSG_GETCLASSID = (CCMSG_BASE + 1)
Global Const CCMSG_GETCLASS = (CCMSG_BASE + 2)
Global Const CCMSG_GETDATA = (CCMSG_BASE + 3)
Global Const CCMSG_GETDATAITEM = (CCMSG_BASE + 4)
Global Const CCMSG_SETCLASS = (CCMSG_BASE + 5)
Global Const CCMSG_SETDATA = (CCMSG_BASE + 6)
Global Const CCMSG_FINDNAME = (CCMSG_BASE + 7)

' Control IDs
Global Const CLASSID_ENHLIST = 9001
Global Const CLASSID_CTREE = 9002
Global Const CLASSID_STYLEBAG = 10101
Global Const CLASSID_IMAGEBAG = 10102
Global Const CLASSID_TSITEM = 20001
Global Const CLASSID_TSLIST = 20002
Global Const CLASSID_TSTREE = 20003
Global Const CLASSID_TABRAS = 20004

' Data IDs
Global Const DATAID_STYLES = 10111
Global Const DATAID_IMAGES = 10112
Global Const DATAID_TSLIST = 10113
Global Const DATAID_TSTREE = 10114
Global Const DATAID_TSITEM = 10115
Global Const DATAID_TABRAS = 10116

' Item Default Indeces (Index for DefImage(I), ...)
Global Const DEFITEM_DISABLED = 0
Global Const DEFITEM_CURRENT = 1
Global Const DEFITEM_SELECTED = 2
Global Const DEFITEM_PARENTC = 3
Global Const DEFITEM_PARENTO = 4
Global Const DEFITEM_LEAF = 5
Global Const NUMDEFITEMS = 6

' Control Flags
'       Border
Global Const BRD_SHADOW = &H1&
Global Const BRD_PUSHED = &H2&
Global Const BRD_BOUTLINE = &H4&
Global Const BRD_BORDER = &H8&
Global Const BRD_BEVEL = &H10&
Global Const BRD_COUTLINE = &H20&
Global Const BRD_ALL = (BRD_SHADOW + BRD_PUSHED + BRD_BOUTLINE + BRD_BORDER + BRD_BEVEL +
BRD_COUTLINE)
'       Scrolling
Global Const VSC_SHOW = &H40&
Global Const VSC_CLICKBORDER = &H80&
Global Const VSC_OVERBORDER = &H100&
Global Const VSC_ALL = (VSC_SHOW + VSC_CLICKBORDER + VSC_OVERBORDER)
Global Const HSC_SHOW = &H200&
Global Const HSC_CLICKBORDER = &H400&
Global Const HSC_OVERBORDER = &H800&
Global Const HSC_ALL = (HSC_SHOW + HSC_CLICKBORDER + HSC_OVERBORDER)
'       Focus Rect
Global Const FOC_TEXT = &H1000&
Global Const FOC_ITEM = &H2000&
Global Const FOC_CONTROL = &H4000&
Global Const FOC_ALL = (FOC_TEXT + FOC_ITEM + FOC_CONTROL)
'       Selection
Global Const SEL_LCLICK = &H8000&
Global Const SEL_RCLICK = &H10000
Global Const SEL_OVER = &H20000
Global Const SEL_MULT = &H40000
Global Const SEL_ALL = (SEL_LCLICK + SEL_RCLICK + SEL_OVER + SEL_MULT)
'       Expand/Collapse
Global Const EXP_LCLICK = &H80000
Global Const EXP_RCLICK = &H100000
Global Const EXP_LDBLCLICK = &H200000
Global Const EXP_RDBLCLICK = &H400000
Global Const EXP_ALL = (EXP_LCLICK + EXP_RCLICK + EXP_LDBLCLICK + EXP_RDBLCLICK)
'       Layout
Global Const LAY_VERT = &H800000
Global Const LAY_HORZ = &H1000000
Global Const LAY_WRAP = &H2000000
Global Const LAY_ALL = (LAY_VERT + LAY_HORZ + LAY_WRAP)
'       Miscellaneous
Global Const CUR_HILITE = &H4000000

' Item Flags
'       Image Position
Global Const IMG_TOP = &H40&
Global Const IMG_LEFT = &H80&
Global Const IMG_RIGHT = &H100&
Global Const IMG_BOTTOM = &H200&
Global Const IMG_ALL = (IMG_TOP + IMG_LEFT + IMG_RIGHT + IMG_BOTTOM)
'       Text
Global Const TXT_WRAP = &H400&
Global Const TXT_LEFT = &H800&
Global Const TXT_RIGHT = &H1000&
Global Const TXT_CENTER = &H2000&
Global Const TXT_THREED = &H4000&
Global Const TXT_SHADOW = &H8000&
Global Const TXT_TRANS = &H10000
Global Const TXT_ALL = (TXT_WRAP + TXT_LEFT + TXT_RIGHT + TXT_CENTER + TXT_THREED + TXT_SHADOW
+ TXT_TRANS)
'       Highlight
Global Const HIL_TEXT = &H20000
Global Const HIL_ITEM = &H40000
Global Const HIL_ALL = (HIL_TEXT + HIL_ITEM)

' Predefined Styles
'       For ControlFlags
Global Const CTL_BORDER = (BRD_BOUTLINE + BRD_BORDER + BRD_BEVEL + BRD_COUTLINE)
Global Const CTL_STANDARD = (VSC_SHOW + HSC_SHOW + FOC_ITEM + SEL_LCLICK + SEL_MULT +
EXP_LDBLCLICK + LAY_VERT + CUR_HILITE)
'       For ItemFlags
Global Const ITM_NORMAL = (IMG_TOP + IMG_LEFT + TXT_LEFT)
Global Const ITM_SELECTED = (ITM_NORMAL + HIL_ITEM)
```

Controls Overview

See also :

Templar List Collection v2.0

TSTree (Full-Featured Tree Control)

The TSTree is a fully-featured, expandable and collapsible hierarchical listbox. By setting the indentation level of items, you can create parent-child relationships in your list. These hierarchical levels can be queried using properties such as ParentIndex, ChildCount, PrevSibIndex, etc.,. Each item in a TSTree provides the full complement of formatting and display possibilities. These include the embedded formatting commands, the ItemFlags, image display using an ImageBag control, and item style effects with a StyleBag. The control itself is fully configurable with 3D borders, shadows, and flags which control item selection, scrollbar display, and auto-expand/collapse. Items can also be individually set to be invisible, selected, or disabled.

TabRas (Tabula Rasa)

The Tabula Rasa control also allows the full flexibilty of control and item display as the TSTree does. Since TabRas stores data as lists of lists, each list is visible to the user at a given time. Using the auto-expand/collapse feature of this control, users can navigate through lists and sub-lists by clicking and double-clicking on items. This is a great tool for any drill-down type of operation. You can also select predefined layouts to arrange your items vertically or horizontally (a scrollable toolbar?), with or without item wrap-around. Tabula Rasa means "Blank Slate" in Latin, so, perhaps the best layout feature in this control is no layout at all! The TabRas control lets you remove any predefined layout, thereby allowing you to set the position, size, image, style, etc. of each item in your list. This control is too cool for words.

TSLabel (Formattable Label/Item)

TSLabel is the odd one out. It's not a list, it's an item! This control mimicks the properties of the items in List Collection listboxes. You have full control over font, color, images, styles, 3D text, shadowed text, etc.,. It can even be used as a button by employing special border flags to produce raised and pressed states. Don't forget, the TSLabel also displays up to 64K of formatted text (scrollbars are included).

ImageBag (List of Images)

The ImageBag provides a source for the images used in the above controls. The ImageBag connects to other controls much in the same way as binding to a data control works; set a property and yoiu're ready to go. Images can be added to and removed from an ImageBag just like items in a listbox. Each image in the list has an ImageName, ImageWidth, ImageHeight, and ImageZoom property. Images can be loaded and saved to disk or copied to and from other controls that also have image properties. Linked controls can access a stored image by specifying either its index in the list or its name.

StyleBag (List of Styles)

The StyleBag holds items which have specific style properties that allow configuration of item font, color, border, etc. By connecting to this control, other List Collection controls can have access to these styles and use them for customizing item display. By combining border and shadow properties, each item can act as button with raised and pressed states. Styles also let you easily indicate the state of list items such as parent items, child items, selected, and disabled items.

CTree (Colorized Tree)

The CTree is a simplified version of the TSTree. It contains nine available "slots" for images that can be displayed with items in the list. Since it is a tree control, it also provides properties for indentation and hierarchical relationships. CTree also supports embedded formatting commands. This makes designing great looking interfaces a snap. We had to put something extra in this control, so, CTree can also serve as a read-only data browsing control. Each record is displayed on a seperate line with fields being displayed according to formatting codes specific to each field.

EnhList (Enhanced List)

This your basic listbox with some extra filing features. Since the EnhList doesn't display images or formatted text, it's simplicity makes it run much faster than the other visible controls in the Collection. It also has properties to display directory listings with all the associated file data such as file size and the date-time stamp. Furthermore, you can even read and write text files to and from the disk. The files contents will be displayed in the EnhList line by line. This is a great way to read and write INI files, log files, etc.

MTree (Memory Tree)

As an invisible version of the TabRas control, the MTree combines true heirarchical data storage without the overhead of a visible control. All of the same tree navigation properties apply to the MTree. You can store all of your data and then traverse it any way you want.

LParse (Line Parser)

The LParse control is an invisible parsing control. This control is unique in that it exposes many of the components used in the parsing process by representing them as different lists. You can add and remove items in the keywords list to have LParse automatically search and identify keywords in the parse string. There's also a list for symbols, whitespace characters, and a list of tokens that result from the parsing operation. The results can be edited a token at a time and then used to generate a new string with the changes intact. The LParse can also automatically parse out numbers and quoted strings as single tokens.

TSTree Control

See also :

Controls Overview

Properties
Methods
Events
Control Flags
Item Flags
Text Formatting Codes

General

TSTree.VBX is a Custom Control for use with VBX-enabled Development Environments.

The TSTree is a Tree control, and functions much in the same way as the CTree control.   That is, it is a listbox whereby an item's indentation level determines its position in a parent-child hierarchy.   The TSTree, however, provides many more features than the CTree control does.

Primarily, TSTree allows you to connect to an ImageBag control.   This allows you to access an almost unlimited number of images of varying size for item display.   The TSTree may also be connected to a StyleBag control to easily display items with different fonts, colors, and sizes.   By connecting to a StyleBag, items can have borders and shadows, as well as some additional text formatting.

The TSTree also has some other features which automate tasks that were painful enough to implement previously. First among these is the auto-expand/collapse of parent items.   You can configure this feature to take place on different types of mouse clicks with the ControlFlags property.   Another powerful feature is the automatic application of styles and images to items that are disabled, current, selected, parent (open and closed), and leaf (no children).   The DefStateImage( ), DefImage( ), DefStyle( ), DefWidth( ), DefHeight( ), and DefFlags( ) properties let you easily set up default looks for certain items.

Features

Memory and Speed

TSTree also provides many features not found in the other controls.   Primarily, the TSTree uses a Virtual Memory subsystem which allows it to potentially access up to 500 Megabytes of RAM.   The BatchMode property also allows you to speed up intensive operations by postponing redundant calculations and paints until the end of the operation.

Borders and Shadows

The Border properties allow you to display ThreeD Borders and Shadows.   You can determine the dimensions of your border by using the BorderWidth, BevelOuter, BevelInner, and ShadowWidth properties.   Customize the look of your border with the BevelLight, BevelDark, ShadowWidth and TrueShadow properties.

Display and Special Effects

This Control also supports Control Flags.   By using the ControlFlags property, you can control the display of the control's border, scrollbars, and focus rectangle.   Certain special effects such as auto-expand/collapse, automatic scrolling, multiple selection, and item layout can also be set with this property.

Akin to Control Flags are Item Flags.   These are flags which you can set for each item to define that item's image placement, text wrapping, text threed and shadow effects, and highlighting.   When connected to a StyleBag control, each item also has the ability to display borders, bevelling, and shadows, also controlled by the ItemFlags( ) and StyleFlags( ) (for the StyleBag) properties.

Data Storage

The TSTree control maintains data in a list.   Each Item has an ItemText property to display formatted text in the display area of the control.   Additional properties include ItemData and ItemString, used to store non-visible data associated with an item.   An Item also has properties for item size, indentation, formating, visibility, and disabling.

Searching

Another set of valuable properties found in this control are the SearchString, FoundIndex, StartSearch, and SearchType properties.   By using these properties, locating data within the control becomes an automated process and, consequently, more efficient than looping through and checking each item yourself.

Enhanced Events

In addition to many of the standard events available with this control, there are several custom events that help you to write efficient and structured code.   The ItemClick, ItemDblClick, and ItemOver events all provide you with information regarding the current selection, button presses, key presses, and mouse location.

TabRas Control

See also :

Controls Overview

Properties
Methods
Events
Control Flags
Item Flags
Text Formatting Codes

General

TabRas.VBX is a Custom Control for use with VBX-enabled Development Environments.

The TabRas is a true Tree control, and functions much in the same way as the MTree control. That is, it is a hierarchical list of nodes, each having a sublist of other nodes, and so on. The TabRas, however, provides many more features than the MTree control does. The TabRas can display its items using several different types of layouts. You can even set the location and size of each item in the list (which is why we named the control Tabula Rasa, "Blank Slate" in Latin).

Primarily, the TabRas control is a visible control whereas the MTree is not. Furthermore, TabRas allows you to connect to an ImageBag control. This allows you to access an almost unlimited number of images of varying size for item display. The TabRas may also be connected to a StyleBag control to easily display items with different fonts, colors, and sizes. By connecting to a StyleBag, items can have borders and shadows, as well as some additional text formatting.

The TabRas also has some other features which automate tasks that were painful enough to implement previously. First among these is the auto-expand/collapse of parent items. You can configure this feature to take place on different types of mouse clicks with the ControlFlags property. Another powerful feature is the automatic application of styles and images to items that are disabled, current, selected, parent (open and closed), and leaf (no children). The DefStateImage( ), DefImage( ), DefStyle( ), DefWidth( ), DefHeight( ), and DefFlags( ) properties let you easily set up default looks for certain items.

Features

Memory and Speed

TabRas also provides many features not found in the other controls. Primarily, the TabRas uses a Virtual Memory subsystem which allows it to potentially access up to 500 Megabytes of RAM. The BatchMode property also allows you to speed up intensive operations by postponing redundant calculations and paints until the end of the operation.

Borders and Shadows

The Border properties allow you to display ThreeD Borders and Shadows. You can determine the dimensions of your border by using the BorderWidth, BevelOuter, BevelInner, and ShadowWidth properties. Customize the look of your border with the BevelLight, BevelDark, ShadowWidth and TrueShadow properties.

Display and Special Effects

This Control also supports Control Flags. By using the ControlFlags property, you can control the display of the control's border, scrollbars, and focus rectangle. Certain special effects such as auto-expand/collapse, automatic scrolling, multiple selection, and item layout can also be set with this property.

Akin to Control Flags are Item Flags. These are flags which you can set for each item to define that item's image placement, text wrapping, text threed and shadow effects, and highlighting. When connected to a StyleBag control, each item also has the ability to display borders, bevelling, and shadows, also controlled by the ItemFlags( ) and StyleFlags( ) (for the StyleBag) properties.

Data Storage

The TabRas control maintains data in a list wherein each item can have child items. Each Item has an ItemText property to display formatted text in the display area of the control. Additional properties include ItemData and ItemString, used to store non-visible data associated with an item. An Item also has properties for item size, formating, visibility, and disabling.

Searching

Another set of valuable properties found in this control are the SearchString, FoundIndex, StartSearch, and SearchType properties. By using these properties, locating data within the control becomes an automated process and, consequently, more efficient than looping through and checking each item yourself.

Enhanced Events

In addition to many of the standard events available with this control, there are several custom events that help you to write efficient and structured code. The ItemClick, ItemDblClick, and ItemOver events all provide you with information regarding the current selection, button presses, key presses, and mouse location.

TSLabel Control

See also :

Controls Overview

Properties
Events
Control Flags
Item Flags
Text Formatting Codes

General

TSLabel.VBX is a Custom Control for use with VBX-enabled Development Environments.

The TSLabel is a Label-Type control, and functions much in the same way as an individual Item works in the TSTree and TabRas controls.

TSLabel allows you to connect to an ImageBag control. This allows you to access an almost unlimited number of images of varying size for item display. The TSLabel may also be connected to a StyleBag control to easily display items with different fonts, colors, and sizes. By connecting to a StyleBag, items can have borders and shadows, as well as some additional text formatting.

Features

Memory and Speed

TSLabel also provides many features not found in the other controls. Primarily, the TSLabel uses a Virtual Memory subsystem which allows it to potentially access up to 500 Megabytes of RAM. The BatchMode property also allows you to speed up intensive operations by postponing redundant calculations and paints until the end of the operation.

Borders and Shadows

The Border properties allow you to display ThreeD Borders and Shadows. You can determine the dimensions of your border by using the BorderWidth, BevelOuter, BevelInner, and ShadowWidth properties. Customize the look of your border with the BevelLight, BevelDark, ShadowWidth and TrueShadow properties.

Display and Special Effects

This Control also supports Control Flags. By using the ControlFlags property, you can control the display of the control's border, scrollbars, and focus rectangle. Certain special effects such as auto-expand/collapse, automatic scrolling, multiple selection, and item layout can also be set with this property.

Akin to Control Flags are Item Flags. These are flags which you can set for each item to define that item's image placement, text wrapping, text threed and shadow effects, and highlighting. When connected to a StyleBag control, each item also has the ability to display borders, bevelling, and shadows, also controlled by the ItemFlags( ) and StyleFlags( ) (for the StyleBag) properties.

Data Storage

The TSLabel control maintains data similar to an Item in a TSTree or TabRas control. An Item has an ItemText property to display formatted text in the display area of the control. Additional properties include ItemData and ItemString, used to store non-visible data associated with an item. An Item also has properties for item size, formating, visibility, and disabling.

Enhanced Events

In addition to many of the standard events available with this control, there are several custom events that help you to write efficient and structured code. The ItemClick, ItemDblClick, and ItemOver events all provide you with information regarding button presses, key presses, and mouse location.

ImageBag Control

See also :

Controls Overview

Properties
Methods
Events
Control Flags

General

ImageBag.VBX is a Custom Control for use with VBX-enabled Development Environments.

The ImageBag is an Image container control, it stores images of varying sizes in a list format that can be easily retreived and used elsewhere in an application.   It can be used independently as a visible display for images and simple text, or as a hidden repository of available images for use with other List Collection Controls.

The ImageBag Control can be used with other List Collection Controls (TabRas, TSLabel, and TSTree) through the use of the ImageBag's hWnd Property.   This property will uniquely identify the ImageBag to the other controls connected to it.   To connect a control to the ImageBag, simply set that control's ImageBag property to the ImageBag's hWnd property (ImageBag.hWnd).

Features

Memory and Speed

ImageBag also provides many features not found in the other controls.   Primarily, the ImageBag uses a Virtual Memory subsystem which allows it to potentially access up to 500 Megabytes of RAM.   The BatchMode property also allows you to speed up intensive operations by postponing redundant calculations and paints until the end of the operation.

Borders and Shadows

The Border properties allow you to display ThreeD Borders and Shadows.   You can determine the dimensions of your border by using the BorderWidth, BevelOuter, BevelInner, and ShadowWidth properties.   Customize the look of your border with the BevelLight, BevelDark, ShadowWidth and TrueShadow properties.

Display and Special Effects

This Control also supports Control Flags.   By using the ControlFlags property, you can control the display of the control's border, scrollbars, and focus rectangle.   Certain special effects such as auto-expand/collapse, automatic scrolling, multiple selection, and item layout can also be set with this property.

Data Storage

The ImageBag control maintains a list of images accessed through the Image( ) property array.   Each Image also has an associated ImageName( ) property, which may be used as a simple (non-formatted) text display or to access that image for use in another control connected to the ImageBag.   The dimensions of each image may be manipulated by setting the ImageWidth( ), ImageHeight( ), and ImageZoom( ) properties.

Searching

Another set of valuable properties found in this control are the SearchString, FoundIndex, StartSearch, and SearchType properties.   By using these properties, locating data within the control becomes an automated process and, consequently, more efficient than looping through and checking each item yourself.

Enhanced Events

In addition to many of the standard events available with this control, there are several custom events that help you to write efficient and structured code.   The ItemClick, ItemDblClick, and ItemOver events all provide you with information regarding the current selection, button presses, key presses, and mouse location.

StyleBag Control

See also :

Controls Overview

Properties
Methods
Events
Control Flags
Item Flags

General

StyleBag.VBX is a Custom Control for use with VBX-enabled Development Environments.

The StyleBag is a Style container control, it stores items with style data in a list format that can be easily retreived and used elsewhere in an application. It can be used independently as a visible display for borders and simple text of different formats, or as a hidden repository of available styles for use with other List Collection Controls.

The type of style data available from a StyleBag includes font information and border information. Within each style, you can set a font's name, size, color, boldness, etc.,. You can also specify border width and color, bevelling, and shadows to be displayed around any item.

The StyleBag Control can be used with other List Collection Controls (TabRas, TSLabel, and TSTree) through the use of the StyleBag's hWnd Property. This property will uniquely identify the StyleBag to the other controls connected to it. To connect a control to the StyleBag, simply set that control's StyleBag property to the StyleBag's hWnd property (StyleBag.hWnd).

Features

Memory and Speed

StyleBag also provides many features not found in the other controls. Primarily, the StyleBag uses a Virtual Memory subsystem which allows it to potentially access up to 500 Megabytes of RAM. The BatchMode property also allows you to speed up intensive operations by postponing redundant calculations and paints until the end of the operation.

Borders and Shadows

The Border properties allow you to display ThreeD Borders and Shadows. You can determine the dimensions of your border by using the BorderWidth, BevelOuter, BevelInner, and ShadowWidth properties. Customize the look of your border with the BevelLight, BevelDark, ShadowWidth and TrueShadow properties.

Display and Special Effects

This Control also supports Control Flags. By using the ControlFlags property, you can control the display of the control's border, scrollbars, and focus rectangle. Certain special effects such as auto-expand/collapse, automatic scrolling, multiple selection, and item layout can also be set with this property.

Akin to Control Flags are Item Flags. These are flags which you can set for each item to define that item's image placement, text wrapping, text threed and shadow effects, and highlighting. When connected to a StyleBag control, each item also has the ability to display borders, bevelling, and shadows, also controlled by the ItemFlags( ) and StyleFlags( ) (for the StyleBag) properties.

Data Storage

The StyleBag control maintains a list of images accessed through the Image( ) property array. Each Image also has an associated ImageName( ) property, which may be used as a simple (non-formatted) text display or to access that image for use in another control connected to the StyleBag. The dimensions of each image may be manipulated by setting the ImageWidth( ), ImageHeight( ), and ImageZoom( ) properties.

Searching

Another set of valuable properties found in this control are the SearchString, FoundIndex, StartSearch, and SearchType properties. By using these properties, locating data within the control becomes an automated process and, consequently, more efficient than looping through and checking each item yourself.

Enhanced Events

In addition to many of the standard events available with this control, there are several custom events that help you to write efficient and structured code. The ItemClick, ItemDblClick, and ItemOver events all provide you with information regarding the current selection, button presses, key presses, and mouse location.

CTree Control

See also :

Controls Overview

Properties
Methods
Events
Text Formatting Codes

General

CTree.VBX is a Custom Control for use with VBX-enabled Development Environments.

The CTree is meant to supplement and/or replace the standard listbox included with Visual Basic. By supporting many of the same standard properties, methods, and events, the CTree may, in a majority of situations, be used as a simple replacement for the standard listbox while capitalizing on some of the advanced features of the CTree.

Features

Memory Management

CTree also provides many features not found in the standard listbox. Primarily, the CTree uses a memory subsystem which allows it to potentially access up to 500 Megabytes of RAM.

Display

CTree also exposes properties enabling a greater control of its inner workings. These include the BatchMode, DrawMethod and HighlightStyle properties. The BatchMode Property allows you to optimize the CTree for speed during intensive operations such as adding many items to the CTree from within a loop or altering the contents of an CTree. The DrawMethod and HighlightStyle properties allow you to customize the display of data within your CTree.

Data Storage

One of the most powerful properties included in the CTree, is the ExtraString property. In many cases, the ItemData property that is associated with each item in a listbox is insufficient to store all the extra data for an item needed in a program. The CTree offers relief in these situations where you need to store data in a listbox but only want to have a portion of it displayed to the user. The ExtraString property may be used in such cases to store all of the data your program needs, yet displaying a separate representation of that data to the user.

Searching

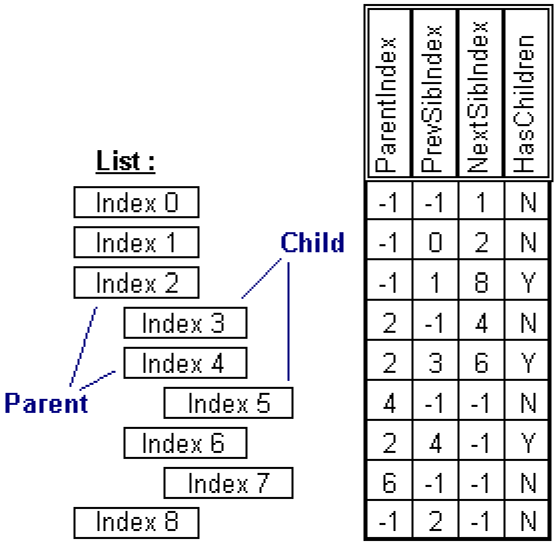
Another set of valuable properties found in the CTree are the SearchString, FoundIndex, StartSearch, and SearchType properties. By using these properties, locating data within the CTree is more automated than looping through and checking each item yourself.

Formatted Display

CTree stands apart from many other types of listboxes by offering properties to display images, multi-colored text, and indention. Displayed text can be formatted by embedding formatting commands into the List( ) Property. Formatting Codes are available to change font attributes, text colors, and text placement.

Tree Manipulation

CTree uses the Indent levels of Items in the List to determine Parent-Child relationships. Properties are available which can retrieve these relationships, as shown below :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

Data Access

By using the BrowseSource and BrowseFields Properties, you can set up the CTree to get a snapshot of data from a Data Control. Once data has been loaded, a program can access a BookMark property for each record loaded.

Enhanced Events

Two events have been added to the CTree which provide a more controlled interaction with the end-user. When using the standard listbox, the Click event is fired whenever the user clicks on an item in the list with the mouse and whenever the ListIndex property is altered from within the code. This may cause confusion in some cases, and may even lead to an infinite loop type of situation called a cascading event. This may occur when you want to change the ListIndex of the listbox from within the Click event (which will call the Click event again, etc.). CTree has the ItemClick and ItemDblClick events which get called only when your user actually clicks on an item (not whenever the ListIndex has been changed). These two events also provide you with information that is difficult to obtain from within the Click event.

EnhList Control

See also :

Controls Overview

Properties
Methods
Events

General

EnhList.VBX is a Custom Control for use with VBX-enabled Development Environments.

The EnhList is meant to supplement and/or replace the standard listbox included with Visual Basic. By supporting many of the same standard properties, methods, and events, the EnhList may, in a majority of situations, be used as a simple replacement for the standard listbox while capitalizing on some of the advanced features of the EnhList.

Features

Memory Management

EnhList also provides many features not found in the standard listbox. Primarily, the EnhList uses a Virtual Memory subsystem which allows it to potentially access up to 500 Megabytes of RAM.

Display

EnhList also exposes properties enabling a greater control of its inner workings. These include the BatchMode, DrawMethod, and HighlightStyle properties. The BatchMode Property allows you to optimize the EnhList for speed during intensive operations such as adding many items to the EnhList from within a loop or altering the contents of an EnhList. The DrawMethod and HighlightStyle properties allow you to customize the display of data within your EnhList.

Data Storage

One of the most powerful properties included in the EnhList, is the ExtraString property. In many cases, the ItemData property that is associated with each item in a listbox is insufficient to store all the extra data for an item needed in a program. The EnhList offers relief in these situations where you need to store data in a listbox but only want to have a portion of it displayed to the user. The ExtraString property may be used in such cases to store all of the data your program needs, yet displaying a separate representation of that data to the user.

Searching

Another set of valuable properties found in the EnhList are the SearchString, FoundIndex, StartSearch, and SearchType properties. By using these properties, locating data within the EnhList is more automated than looping through and checking each item yourself.

File Viewing

The EnhList a ListDir Property which will list all files and directories from a path much like the Standard Dir and File listboxes. FileName and FileAction Properties are available to Read, Write, and Delete text files on disk.

Enhanced Events

Two events have been added to the EnhList which provide a more controlled interaction with the end-user. When using the standard listbox, the Click event is fired whenever the user clicks on an item in the list with the mouse and whenever the ListIndex property is altered from within the code. This may cause confusion in some cases, and may even lead to an infinite loop type of situation called a cascading event. This may occur when you want to change the ListIndex of the listbox from within the Click event (which will call the Click event again, etc.). EnhList has the ItemClick and ItemDblClick events which get called only when your user actually clicks on an item (not whenever the ListIndex has been changed). These two events also provide you with information that is difficult to obtain from within the Click event.

MTree Control

See also :

Controls Overview

Properties
Methods

General

MTree.VBX is a Custom Control for use with VBX-enabled Development Environments.

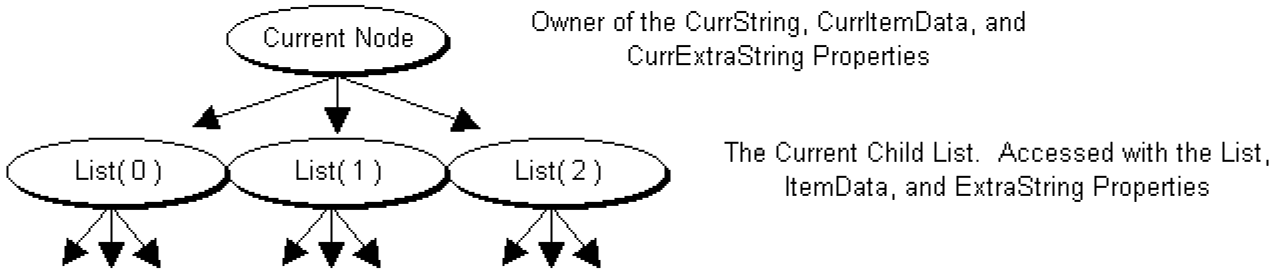
The MTree is designed to be an easy to use, yet powerful Tree control. All Tree operations are performed through the use of properties.

Features

Memory Management

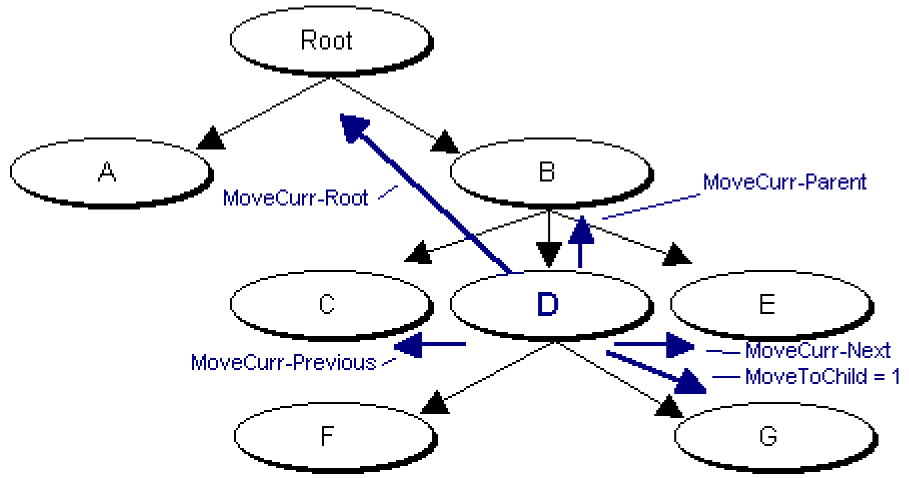
MTree also provides many features not found in the standard listbox. Primarily, the MTree uses a memory subsystem which allows it to potentially access up to 500 Megabytes of RAM.

Data Storage


Current Node
Owner of the CurrString, CurrItemData, and CurrExtraString Properties
List(0)
List(1)
List(2)
The Current Child List. Accessed with the List, ItemData, and ExtraString Properties

To simplify the tree concept, the MTree control can be viewed as being a List in which each Item has a sublist of it's own. Each Item in a sublist may have another sublist of it's own, and so on. In fact, the MTree control shares many of the List (ListCount, ListIndex, NewIndex) and Item (List( ), ItemData( ), ExtraString( )) Properties found in the other List Controls. When used with MTree, these properties refer to the sublist of the Current Node.

Tree Traversal



An MTree Control always has a Current Node. The Current Node can be uniquely identified by the CurrPath Property. When an MTree Control is initialized, it starts out with a Root Node. This Root Node can never be deleted and becomes the Current Node until the Current Node is changed, at Run-Time, with the CurrPath, MoveCurr or MoveToChild Properties.

Searching

Mtree also offers a set searching properties to find item values in a sublist. These are the SearchString, FoundIndex, StartSearch, and SearchType properties. By using these properties, locating data within the MTree is more automated than looping through and checking each item yourself.

LParse Control

See also :

Controls Overview

Properties
Methods

General

LParse.VBX is a Custom Control for use with VBX-enabled Development Environments.

The LParse Control is unique in that it provides a very convinient and efficient means of parsing string oriented data. LParse is also very configurable, and can be applied in a variety of situations.

Features

Memory Management

As with the other Controls in this package, LParse also has access to an excessive amount of storage space. There will rarely be times when you will need to count on this amount of memory, but it is always nice to never have to worry about it either.

Configurable Parsing

LParse can be configured to handle virtually any type of parsing operation. The LParse can automatically identify Whitespace characters (as specified by the WSpaceChars Property), Keywords, Symbols (by filling the Keywords and Symbols Lists - see ListType Property), single and double quoted strings (bSingleQuotes and bDoubleQuotes), and numerics (bDetectInteger and bDetectReal).

Searching

Another set of valuable properties found in the LParse are the SearchString, FoundIndex, StartSearch, and SearchType properties. By using these properties, locating data within the LParse is more automated than looping through and checking each item yourself.

Events Overview

See also :

Templar List Collection v2.0

Standard Events
ItemClick Event
ItemDblClick Event

Standard Events

See also :

Events Overview

Events

Click

DblClick

DragDrop

DragOver

GotFocus

KeyDown

KeyPress

KeyUp

LostFocus

MouseDown

MouseMove

MouseUp

ItemClick Event

See also :

Events Overview

Applies To

CTree

EnhList

ImageBag

StyleBag

TabRas

TSLabel

TSTree

Event Arguments

```
(ItemIndex As Integer, BtnClicked As Integer, BtnState As Integer, KeyState As Integer, X As
Single, Y As Single)
```

- **ItemIndex - Index of the Item Clicked on by the User**
- **BtnClicked - Indicates which Button was Clicked**

1 = Left Button, 2 = Right Button, 4 = Middle Button

- **BtnState - Indicates the state of the MouseButtons during the Click**

1 = Left Button, 2 = Right Button, 4 = Middle Button

- **KeyState - Indicates the state of the Shift and Control Keys during the Click**

1 = Shift Key, 2 = Control Key

- **X - The X Coordinate of the Click**
- **Y - The Y Coordinate of the Click**

Description

The ItemClick Event, analogous to the Standard Click Event, is fired when an end-user clicks on an item in the List; However, unlike the Standard Click Event, the ItemClick Event is not fired whenever the ListIndex Property is changed.

The Standard Click Event fires whenever the ListIndex changes (by clicking a different item in the list, or when set within VB code). This side-effect of the Standard Click Event makes it difficult to code sometimes, as you don't know why the event is being called.

Purpose

This event is most useful when you wish to detect only interaction from the user. The ItemIndex argument informs you of the actual item that was clicked. Note that when this event is called, you are guaranteed to have some items in your list and that the user has actually selected one of them. This will save you the code to do this checking yourself.

Example

```
Sub List_ItemClick( ItemIndex As Integer, BtnClicked As Integer, BtnState As Integer,
KeyState As Integer, X As Single, Y As Single )

  Dim S As String
  ' Only Process a Left Click
  If BtnClicked <> 1 Then Exit Sub
  ' Get Item Info
  S = "Left-Click Info :" + Chr$(13)
  S = S + ("Index = " + Str$(ItemIndex) + Chr$(13))
  S = S + ("Text = " + List.List(ItemIndex) + Chr$(13))
  S = S + ("X = " + Str$(X) + ", Y = " + Str$(Y) + Chr$(13))
  ' Check if Right Button is Also Down
  If (BtnState And 2) = 2 Then
    S = S + ("Right Button is Down" + Chr$(13))
  End If
  ' Check Keys
  If (KeyState And 1) = 1 Then
    S = S + ("Shift Key is Pressed" + Chr$(13))
  End If
  If (KeyState And 2) = 2 Then
    S = S + ("Control Key is Pressed" + Chr$(13))
  End If
 ' Display Results
  MsgBox S

End Sub
```

ItemDblClick Event

See also :

Events Overview

Applies To

CTree

EnhList

ImageBag

StyleBag

TabRas

TSLabel

TSTree

Event Arguments

```
(ItemIndex As Integer, BtnClicked As Integer, BtnState As Integer, KeyState As Integer, X As
Single, Y As Single)
```

- **ItemIndex - Index of the Item Clicked on by the User**
- **BtnClicked - Indicates which Button was Clicked**

1 = Left Button, 2 = Right Button, 4 = Middle Button

- **BtnState - Indicates the state of the MouseButtons during the Click**

1 = Left Button, 2 = Right Button, 4 = Middle Button

- **KeyState - Indicates the state of the Shift and Control Keys during the Click**

1 = Shift Key, 2 = Control Key

- **X - The X Coordinate of the Click**
- **Y - The Y Coordinate of the Click**

Description

The ItemDblClick Event, analogous to the Standard Double-Click Event, is fired when an end-user double-clicks on an item in the List.

Purpose

This event is most useful when you wish to detect only interaction from the user.   The ItemIndex argument informs you of the actual item that was double-clicked.   Note that when this event is called, you are guaranteed to have some items in your list and that the user has actually selected one of them.   This will save you the code to do this checking yourself.

Example

```
Sub List_ItemClick( ItemIndex As Integer, BtnClicked As Integer, BtnState As Integer,
KeyState As Integer, X As Single, Y As Single )

  Dim S As String
  ' Only Process a Left Click
  If BtnClicked <> 1 Then Exit Sub
  ' Get Item Info
  S = "Left-DoubleClick Info :" + Chr$(13)
  S = S + ("Index = " + Str$(ItemIndex) + Chr$(13))
  S = S + ("Text = " + List.List(ItemIndex) + Chr$(13))
  S = S + ("X = " + Str$(X) + ", Y = " + Str$(Y) + Chr$(13))
  ' Check if Right Button is Also Down
  If (BtnState And 2) = 2 Then
    S = S + ("Right Button is Down" + Chr$(13))
  End If
  ' Check Keys
  If (KeyState And 1) = 1 Then
    S = S + ("Shift Key is Pressed" + Chr$(13))
  End If
  If (KeyState And 2) = 2 Then
    S = S + ("Control Key is Pressed" + Chr$(13))
  End If
 ' Display Results
  MsgBox S

End Sub
```

Methods Overview

Templar List Collection v2.0

Applies To

CTree
EnhList
ImageBag
StyleBag
TabRas
TSTree
MTree
LParse

Methods

AddItem
RemoveItem
Clear
Refresh

Properties Overview

Templar List Collection v2.0

- CTree Properties
- EnhList Properties
- ImageBag Properties
- StyleBag Properties
- TSLabel Properties
- TabRas Properties
- TSTree Properties
- MTree Properties
- LParse Properties

Property Summaries

- Control Properties
- Control Border Properties
- List Display Properties
- List Data Properties
- List Searching Properties
- Item Display Properties
- Item Defaults Properties
- Item Data Properties
- Image Properties
- Style Properties
- Tree Properties
- Tree Path Properties
- File/Dir View Properties
- Data Browsing Properties
- Parser Properties
- Other Properties

Control Properties

Properties Overview

									Property Name	Short Description (Click Property Name for Detail)
x	x	x	x	x	x	x	x	x	hWnd Property	Window handle of control
x	x	x	x	x	x	x	x	x	BatchMode Property	Temporarily disable output display to optimize speed
x						x	x	x	EscChar Property	Character to use with Text Formatting Codes
		x			x	x	x	x	PopUp Property	Sets control as popup window outside of Form
						x	x	x	ImageBag Property	hWnd of a source ImageBag
						x	x	x	StyleBag Property	hWnd of a source StyleBag
		x			x	x	x	x	ControlFlags Property	Set borders, shadow, selection, and scroll-bars for control

Control Border Properties

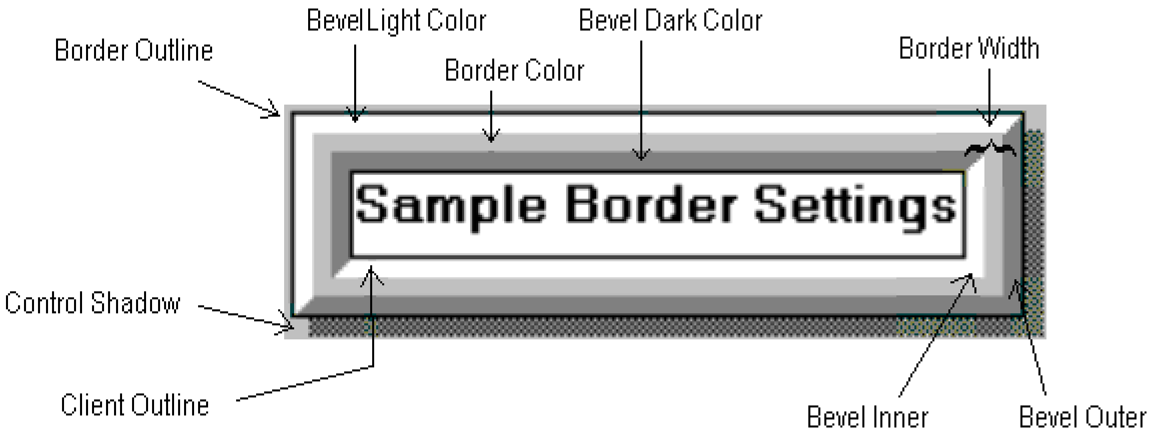
Properties Overview

									Property Name	Short Description (Click Property Name for Detail)
		x			x	x	x	x	BorderWidth	Width of the controls border (including the bevelling)
		x			x	x	x	x	BorderColor	Color of the controls border
		x			x	x	x	x	BevelOuter	Amount to bevel the outside of the controls border
		x			x	x	x	x	BevelInner	Amount to bevel the inside of the controls border
		x			x	x	x	x	BevelLight	Color of edges facing towards the light source
		x			x	x	x	x	BevelDark	Color of edges facing away from the light source
		x			x	x	x	x	ShadowWidth	Width of controls drop shadow
		x			x	x	x	x	TrueShadow	Draw controls drop shadow with a transparent background

Many of the Templar List Collection controls have properties for customizing the border of the control. There are esentially two categories of border properties :

- Widths : **BorderWidth**, **BevelInner**, **BevelOuter**, **ShadowWidth**
- Colors : **BorderColor**, **BevelLight**, **BevelDark**, **TrueShadow**

When using these properties, it is important to note that the border element does not become visible until the corresponding **ControlFlag** for that element is turned on. This allows you to easily implement special effects (border is bevelled when control has focus, control can have a button-like pushable effect, etc.) in response to certain events. Rather than setting and clearing each border element property, the **ControlFlags** will turn them on and off.



List Display Properties

Properties Overview

									Property Name	Short Description (Click Property Name for Detail)
x									DisplayType	Display text as Regular, Formatted, or RawText
x	x								DrawMethod	Display list with Normal or Smooth scrolling
x	x								HighlightStyle	Highlight style for the current item (None, Text, or Item)
x	x								ScrollWidth	Set scrollable width for control
x						x		x	IndentWidth	Amount to indent an item for each indent level
		x			x	x		x	ItemSpacing	Spacing to add to top and left of items
						x			MinScroll	Minimum amount to scroll horizontally and vertically
						x		x	ColumnSpacing	Spacing to use between columns of items
x	x								TitleText	Title text displayed at top of control

List Data Properties

Properties Overview

									Property Name	**Short Description** (Click Property Name for Detail)
x	x	x	x	x	x	x		x	ListCount	Total number of items that have been added to a list
x	x	x	x	x	x	x		x	ListIndex	Current item in list
x	x	x	x	x	x	x		x	NewIndex	Index of most recently added item in list
x	x	x			x	x		x	TopIndex	First visible item in list display
		x			x	x		x	BottomIndex	Last visible item at bottom of list display

List Searching Properties

Properties Overview

					ABC		TS		Property Name	Short Description (Click Property Name for Detail)
x	x	x	x	x	x	x		x	StartSearch	Iindex of first item to start with in the search
x	x	x	x	x	x	x		x	SearchType	Type of data to search for (Text, ItemData, ExtraString)
x	x	x	x	x	x	x		x	SearchString	Value to search for in list
x	x	x	x	x	x	x		x	FoundIndex	Index of matched item (-1 if no match)

The List Searching properties search all the items of a list for specific values. By supplying a value to search for with the **SearchString** property, the list will begin to automatically search all of its items for that string. If a match is found, the **FoundIndex** property will contain the index of the Item that was matched (-1 if there was no match found). The **StartSearch** property can be used to designate a starting point for searches. This is helpful if you need to find more than one instance of a string within the list.

A search can be performed for matching values to be found in either the List/Text, ExtraString, or ItemData properties of an item. The **SearchType** property allows you to define which Item property will be searched for matches, as follows :

0	List( ) / ItemText( ) property
1	ItemData( ) property
2	ExtraString( ) property

Here is an example which finds and counts all instances of particular value :

```
Function CountAll( lst as CTree, ByVal s as String ) as Long
        Dim I as Long
        CountAll = 0
        While True
                 Do Search
                lst.StartSearch = 0
                lst.SearchString = s
                 Check if Not Found
                If( lst.FoundIndex < 0 ) Then Exit Function
                ' Add Match to Count
                CountAll = CountAll + 1
                ' Check if Done with List
                If( lst.FoundIndex = (lst.ListCount - 1) ) Then Exit Function
                ' Setup for Next Search - StartSearch with Next Item
                lst.StartSearch = (lst.FoundIndex + 1)
        Wend
End Function
```

Item Display Properties

Properties Overview

									Property Name	**Short Description** (Click Property Name for Detail)
x								x	ItemIndent( )	Indent level of an item
x						x	x	x	ItemImage( )	Image Name or Number for item
						x	x	x	ItemStyle( )	Style Name or Number for item
						x		x	ItemStateImage( )	Image displayed outside and to the left of an item
						x	x	x	ItemFlags( )	Set flags to display special effects within an item
						x		x	ItemVisible( )	Set an items visiblity on or off
						x		x	ItemEnabled( )	Enable or disable an item
						x		x	ItemSelected( )	Select or unselect item
						x	x	x	ItemHeight( )	Height (in pixels) of item
x									ItemHeight	Height (in pixels) of item
						x			ItemTop( )	Top of item (in virtual coords)
						x			ItemLeft( )	Left of item (in virtual coords)
						x	x	x	ItemWidth( )	Width of item

Item Defaults Properties

Properties Overview

		Property Name	**Short Description** (Click Property Name for Detail)
x	x	DefStateImage( )	Default ItemStateImage for item
x	x	DefImage( )	Default ItemImage for Item
x	x	DefStyle( )	Default ItemStyle for item
x	x	DefWidth( )	Default ItemWidth for item
x	x	DefHeight( )	Default ItemHeight for item
x	x	DefFlags( )	Default ItemFlags for item

The Item Defaults properties allow you to setup default characteristics for items in your list.   Each of these properties acts as an array where the index used in the property specifies one of :

0	**Disabled items**
1	**Current item (identified by ListIndex)**
2	**Selected items**
3	**Closed parent items (has children)**
4	**Open parent items (has children)**
5	**Leaf items (no children)**

The control uses these defaults to fill in any missing information about an items appeareance when it is trying to draw and size that item.

These properties are only applied to an item if there are no corresponding values for these properties available to the item itself.   This allows you to set global charecteristics for different types of items.   You can still customize a particular item by setting that items image, style, etc. properties, thereby overiding the defaults.

Actually, the default highlighting mechanism for the control is enabled with the Item Defaults properties.   When a new control is created, the following defaults are set :

```
TSTree1.DefFlags(1) = HIL_ITEM            Highlight whole item for current item
TSTree1.DefFlags(2) = HIL_ITEM            Highlight whole item for selected items
```

For example, to display separate images for open parents and for closed parents :

```
TSTree1.ImageBag = ImageBag1.hWnd         Attach to source of images
TSTree1.DefImage(3) = Closed.BMP   Image for closed parents
TSTree1.DefImage(4) = Open.BMP            Image for open parents
TSTree1.DefImage(5) = Child.BMP           Image for leaf item
```

Item Data Properties

Properties Overview

									Property Name	**Short Description** (Click Property Name for Detail)
x	x		x	x		x	x	x	ItemData ( )	Long value associated with each item
						x	x	x	ItemString ( )	String value associated with each item
						x	x	x	ItemText ( )	Text to display in listbox for item
x	x	x	x	x	x		x	x	List ( )	Text to display in listbox for item (Alias for ItemText( ))
x						x		x	RawText ( )	ItemText stripped of formatting codes
x	x		x	x				x	ExtraString ( )	String value associated with each item (Alias for ItemString)
				x					CurrString	Current Nodes associated Text (Alias for CurrItemText)
				x					CurrExtraString	Current Nodes associated String value (Alias for CurrItemString)
				x					CurrItemData	Current Nodes associated Long value
						x	x	x	ItemRawText	Current Nodes associated Long value

Image Properties

Properties Overview

		Property Name	**Short Description** (Click Property Name for Detail)
	x	ImageName ( )	Name used to identify image
	x	Image ( )	Image property for item
	x	ImageWidth ( )	Width of image
	x	ImageHeight ( )	Height of image
	x	ImageZoom ( )	Image zoom factor
x		ImageWidth	Width of images displayed in items
x		Image 1..9	Images used for display with items

Style Properties

Properties Overview

	Property Name	**Short Description** (Click Property Name for Detail)
x	StyleName ( )	Name used to identify style
x	StyleFontName ( )	FontName for style
x	StyleFontSize ( )	FontSize for style
x	StyleBold ( )	FontBold for style
x	StyleItalic ( )	FontItalic for style
x	StyleUnderline ( )	FontUnderline for style
x	StyleStrikethru ( )	FontStrikethru for style
x	StyleBackColor ( )	BackColor for style
x	StyleForeColor ( )	ForeColor for style
x	StyleWidth ( )	Width of item
x	StyleHeight ( )	Height of item
x	StyleBorderWidth ( )	Width of items border
x	StyleBorderColor ( )	Color of the items border
x	StyleBevelInner ( )	Amount to bevel the inside of the items border
x	StyleBevelOuter ( )	Amount to bevel the outside of the items border
x	StyleBevelLight ( )	Color of edges facing towards the light source
x	StyleBevelDark ( )	Color of edges facing away from the light source
x	StyleShadowWidth ( )	Width of items drop shadow
x	StyleFlags ( )	ItemFlags for item

Tree Properties

Properties Overview

				Property Name	**Short Description** (Click Property Name for Detail)
x			x	ParentIndex ( )	Index of parent item
x			x	PrevSibIndex ( )	Index of previous sibling item
x			x	NextSibIndex ( )	index of next sibling item
x				HasChildren ( )	Indicates of an item has child items
	x	x	x	ChildCount ( )	Number of child items for a parent item
		x	x	ChildrenVisible ( )	Indicates if child items are visible
	x	x		MoveCurr	Moves the current node
	x	x		MoveToChild	Moves the curent node to a chld node
	x	x		BadMove	Indicates if the last MoveCurr or MoveToChild resulted in error
	x	x		CurrDepth	Indicates the depth of the current node

Tree Path Properties

Properties Overview

					Property Name	**Short Description** (Click Property Name for Detail)
x			x	x	PathChar	Character used to separate items in a path string
x				x	AbsPath ( )	Items path constructed with items index
x				x	RelPath ( )	Items path constructed with items text
		x	x		CurrPath	Move to the item specified by an AbsPath string

File/Dir Viewing Properties

Properties Overview

	Property Name	**Short Description** (Click Property Name for Detail)
x	FileExists	Determine if FileName exists
x	FileAction	Load, Save, or Delete FileName
x	FileName	FileName used for FileExists and FileAction
x	ListDir	Directory path (with wildcards) to list

Data Browsing Properties

Properties Overview

	Property Name	Short Description (Click Property Name for Detail)
x	BrowseSource	Name of source Data Control to use for browsing
x	BrowseFields	Names of fields (from BrowseSource) to display
x	BrowseRefresh	Refresh the contents of the list from the BrowseSource
x	BookMark ( )	BookMark value for a particular record from BrowseSource

Parsing Properties

Properties Overview

	Property Name	**Short Description** (Click Property Name for Detail)
x	bCaseSensitive	Control case-sensitivity when parsing
x	bCaseEnforce	Force case of parsed Keywords and Symbols
x	bDoubleQuotes	Treat double-quoted strings as a single token
x	bSingleQuotes	Treat single-quoted strings as a single token
x	bDetectInteger	Treat integer numbers as a single token
x	bDetectReal	Treat real numbers as a single token
x	ListType	Current working list (Keywords, Symbols, Parsed Tokens)
x	MaxTokenLen	Max length of parsed token
x	ParseString	String to parsed
x	TokenType ( )	Identifies type of token (Identifier, Keyword, etc.)
x	TokenNum ( )	Identify exactly which token was found
x	WSpaceChars	Identifies whitespace chars (i.e. blank chars)

Other Properties

Properties Overview

									Property Name	**Short Description** (Click Property Name for Detail)
x	x	x	x	x	x	x	x	x	About	Displays About Box
x	x	x			x	x	x	x	ClonePropsGet	Copy property settings from another contol
x	x	x			x	x	x	x	ClonePropsSet	Copy property settings to another contol
		x			x	x	x	x	ControlVersion	Loaded version of control

Ctree Properties

About
AbsPath( )
BackColor
BackgroundImage
BatchMode
BookMark
BrowseFields
BrowseRefresh
BrowseSource
ClonePropsGet
ClonePropsSet
DataSource
DataField
DataChanged
DisplayType
DragIcon
DragMode
DrawMethod
Enabled
EscChar
ExtraString( )
FontBold
FontItalic
FontName
FontSize
FontStrikethru
FontUnderline
ForeColor
FoundIndex
HasChildren( )
Height
HelpContextID
HighlightStyle
Hwnd
Image1...9
ImageWidth
IndentWidth
Index
ItemData( )
ItemHeight
ItemImage( )
ItemIndent( )
Left
List( )
ListCount
ListIndex
MousePointer
Name
NewIndex
NextSibIndex( )
ParentIndex( )
PathChar
PrevSibIndex( )
RawText( )
RelPath( )
ScrollWidth
SearchString
SearchType
StartSearch
TabIndex
TabStop
Tag
TitleText
Top
TopIndex
Visible
Width

EnhList Properties

About	Align	BackColor	BatchMode
BorderStyle	ClipControls	ClonePropsGet	ClonePropsSet
DragIcon	DragMode	DrawMethod	Enabled
ExtraString( )	FileAction	FileExists	FileName
FontBold	FontItalic	FontName	FontSize
FontStrikeThru	FontUnderline	ForeColor	FoundIndex
Height	HelpContextID	HighlightStyle	Hwnd
Index	ItemData( )	Left	List( )
ListCount	ListDir	ListIndex	Mousepointer
Name	NewIndex	ScrollWidth	SearchType
SearchString	StartSearch	TabIndex	TabStop
Tag	TitleText	Top	TopIndex
Visible	Width		

ImageBag Properties

About
Align
BackColor
Batchmode
BevelDark
BevelInner
BevelLight
BevelOuter
BorderColor
BorderWidth
BottomIndex
ClipControls
ClonePropsGet
ClonePropsSet
ControlFlags
ControlVersion
DragIcon
DragMode
Enabled
FontBold
FontItalic
FontName
FontSize
FontStrikeThru
FontUnderline
FoundIndex
ForeColor
Height
HelpContextID
Hwnd
Index
Image( )
ImageHeight( )
ImageName( )
ImageWidth( )
ImageZoom( )
ItemSpacing
Left
List( )
ListCount
ListIndex
MousePointer
Name
NewIndex
PopUp
SearchString
SearchType
ShadowWidth
StartSearch
TabIndex
TabStop
Tag
Top
TopIndex
TrueShadow
Visible
Width

StyleBag Properties

About

Align

BackColor

BatchMode

BevelDark

BevelInner

BevelLight

BevelOuter

BorderColor

BorderWidth

BottomIndex

ClipControls

ClonePropsGet

ClonePropsSet

ControlFlags

ControlVersion

DragIcon

DragMode

Enabled

FontBold

FontItalic

FontName

FontSize

FontStrikeThru

FontUnderline

ForeColor

FoundIndex

Height

HelpContextID

Hwnd

Index

ItemSpacing

Left

List( )

ListCount

ListIndex

MousePointer

Name

NewIndex

PopUp

SearchString

SearchType

ShadowWidth

StartSearch

StyleBackColor( )

StyleBevelDark( )

StyleBevelInner( )

StyleBevelLight( )

StyleBevelOuter( )

StyleBold( )

StyleBorderColor( )

StyleBorderWidth( )

StyleFlags( )

StyleFontName( )

StyleFontSize( )

StyleForeColor( )

StyleHeight( )

StyleItalic( )

StyleName( )

StyleShadowWidth( )

StyleStrikeThru( )

StyleUnderline( )

StyleWidth( )

TabIndex

TabStop

Tag

Top

TopIndex

TrueShadow

Visible

Width

TSLabel Properties

- About
- Align
- BackColor
- BatchMode
- BevelDark
- BevelInner
- BevelLight
- BevelOuter
- BorderColor
- BorderWidth
- ClipControls
- ClonePropsGet
- ClonePropsSet
- ControlFlags
- ControlVersion
- DragIcon
- DragMode
- Enabled
- EscChar
- FontBold
- FontItalic
- FontName
- FotSize
- FontStrikeThru
- FontUnderline
- ForeColor
- Height
- HelpContextID
- Hwnd
- ImageBag
- Index
- ItemData
- ItemFlags
- ItemHeight
- ItemImage
- ItemRawText
- ItemString
- ItemStyle
- ItemText
- ItemWidth
- Left
- MousePointer
- Name
- PopUp
- ShadowWidth
- StyleBag
- TabIndex
- TabStop
- Tag
- Top
- TrueShadow
- Visible
- Width

TabRas Properties

About
Align
BackColor
BadMove
BatchMode
BevelDark
BevelInner
BevelLight
BevelOuter
BorderColor
BorderWidth
BottomIndex
ChildCount( )
ChildrenVisible( )
ClipControls
ClonePropsGet
ClonePropsSet
ColumnSpacing
ControlFlags
ControlVersion
CurrDepth
CurrIndex
CurrPath
DefFlags
DefHeight
DefImage
DefStateImage
DefStyle
DefWidth
DragIcon
DragMode
Enabled
EscChar
FontBold
FontItalic
FontName
FontSize
FontStrikeThru
FontUnderline
ForeColor
FoundIndex
Height
HelpContextID
Hwnd
ImageBag
IndentWidth
Index
ItemData( )
ItemEnabled( )
ItemFlags( )
ItemHeight( )
ItemImage( )
ItemLeft( )
ItemRawText( )
ItemSelected( )
ItemSpacing
ItemStateImage( )
ItemString( )
ItemStyle( )
ItemText( )
ItemTop( )
ItemVisible( )
ItemWidth( )
Left
List( )
ListCount
ListIndex
MinScroll
MousePointer
MoveCurr
MoveToChild
Name
NewIndex
PathChar
PopUp
RawText( )
SearchString
SearchType
ShadowWidth
StartSearch
StyleBag
TabIndex
TabStop
Tag
Top
TopIndex
TrueShadow
Visible
Width

TSTree Properties

- About
- AbsPath( )
- Align
- BackColor
- BatchMode
- BevelDark
- BevelInner
- BevelLight
- BevelOuter
- BorderColor
- BorderWidth
- BottomIndex
- ChildCount( )
- ChildrenVisible( )
- ClonePropsGet
- ClonePropsSet
- ControlFlags
- ControlVersion
- DefFlags
- DefHeight
- DefImage
- DefStateImage
- DefStyle
- DefWidth
- DragIcon
- DragMode
- Enabled
- EscChar
- ExtraString( )
- FontBold
- FontItalic
- FontName
- FontSize
- FontStrikeThru
- FontUnderline
- ForeColor
- FoundIndex
- Height
- HelpContextID
- Hwnd
- ImageBag
- IndentWidth
- Index
- ItemData( )
- ItemEnabled( )
- ItemFlags( )
- ItemHeight( )
- ItemImage( )
- ItemIndent( )
- ItemRawText( )
- ItemSelected( )
- ItemSpacing
- ItemStateImage( )
- ItemString( )
- ItemStyle( )
- ItemText( )
- ItemVisible( )
- ItemWidth( )
- Left
- List( )
- ListCount
- ListIndex
- MousePointer
- Name
- NewIndex
- NextSibIndex( )
- ParentIndex( )
- PathChar
- PopUp
- PrevSibIndex( )
- RawText( )
- RelPath( )
- SearchString
- SearchType
- ShadowWidth
- StartSearch
- StyleBag
- TabIndex
- TabStop
- Tag
- Top
- TopIndex
- TrueShadow
- Visible
- Width

MTree Properties

- About
- BadMove
- BatchMode
- ChildCount( )
- CurrDepth
- CurrExtraString
- CurrItemData
- CurrPath
- CurrString
- ExtraString( )
- FoundIndex
- HelpContextID
- Hwnd
- Index
- ItemData( )
- Left
- List( )
- ListCount
- ListIndex
- MoveCurr( )
- MoveToChild( )
- Name
- NewIndex
- SearchString
- SearchType
- StartSearch
- Tag
- Top

LParse Propeties

About
BatchMode
bCaseEnforce
bCaseSensitive
bDetectInteger
bDetectReal
bDoubleQoutes
bSingleQoutes
ExtraString( )
FoundIndex
HelpContextID
Hwnd
Index
ItemData( )
Left
List( )
ListCount
ListIndex
ListType
MaxTokenLen
Name
NewIndex
ParseString
SearchString
SearchType
StartSearch
Tag
TokenNum( )
TokenType( )
Top
WSpaceChars

Text Formatting Codes Example

Note: Using formatting codes embedded into the text that is displayed in the control can enhance readability. Using the RawText( ) property, you can get the text back out of the list without the formatting codes.

Font Attributes and Colors:

Joe Mankuso has a red dog.

\CFBLU\\BJoe Mankuso\b\CFBLK\ has a \CFRED\red \CFBLK\ dog.

Tab over to characters (appearance of columns):

Jane Smith Female 25

John Doe Male 31

Jane Smith \TC16\\CFBLU\Female\TC26\25

John Doe \TC16\\CFBLU\Male\TC26\31

In a loop:

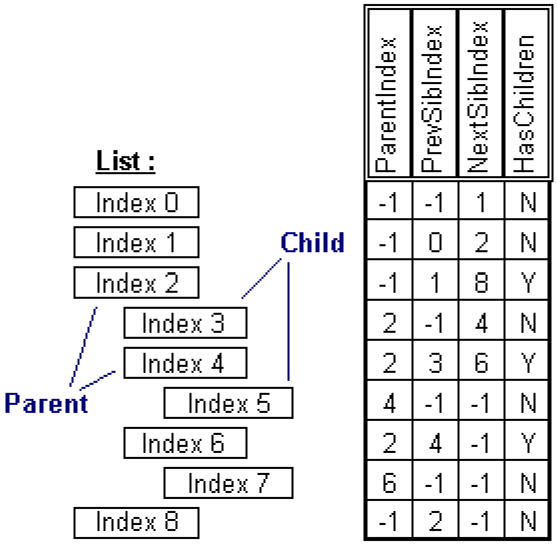
```
For I = 0 to (DS!RecordCount -1)
   lst.additem \I & DS!Name & \I & \TC16\\CFBLU\ & DS!Gender & \TC26\ & DS!Age
   DS.MoveNext
next i
```

The About Property

See also :

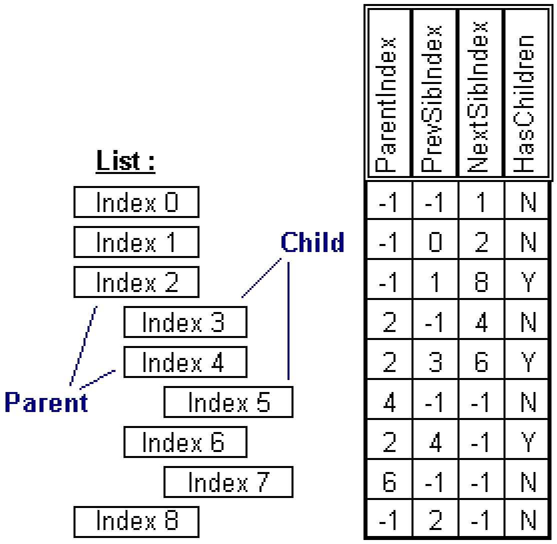
Other Properties

Applies To :



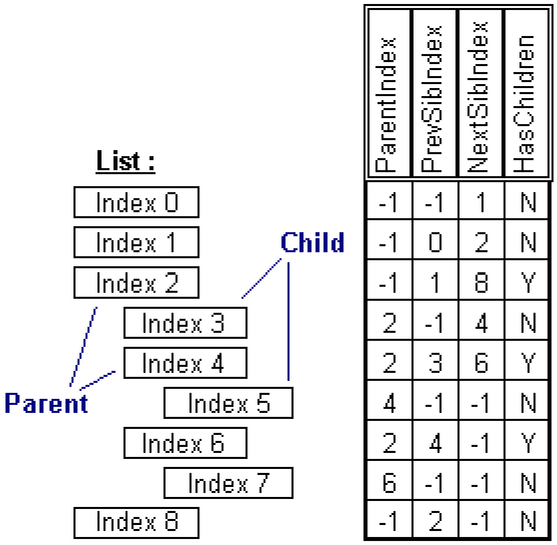
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



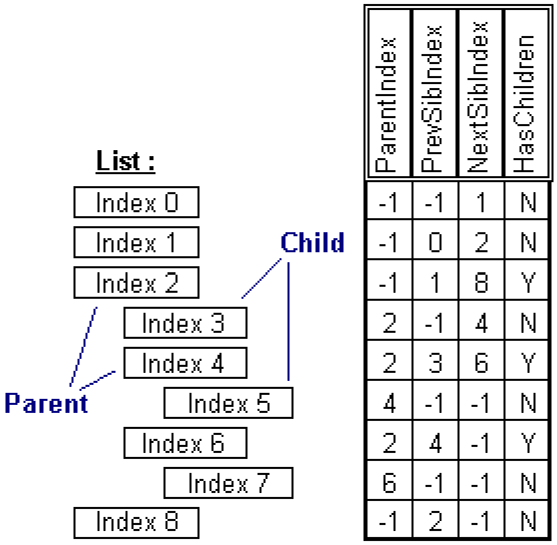
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList



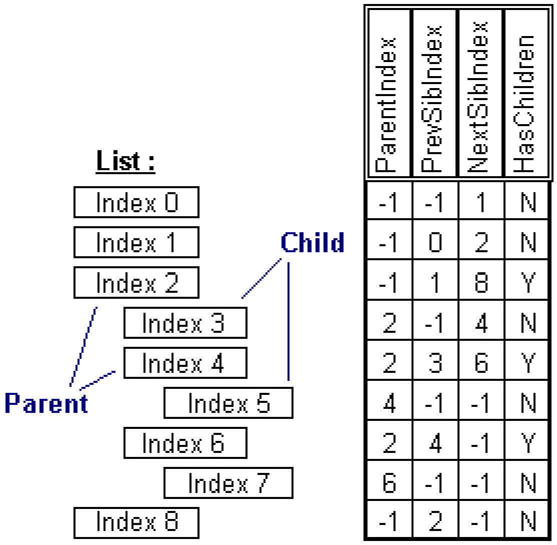
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



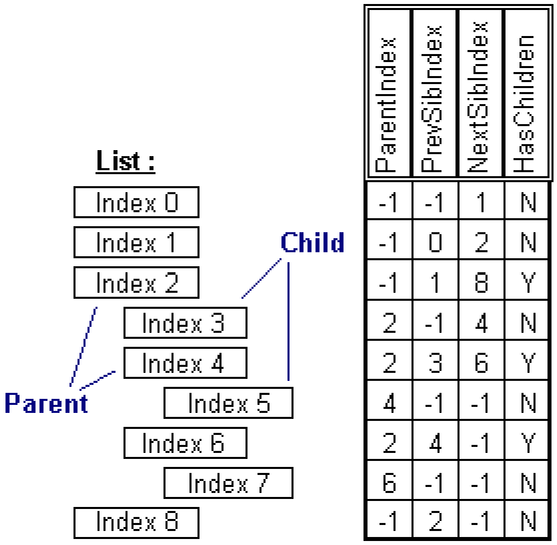
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse



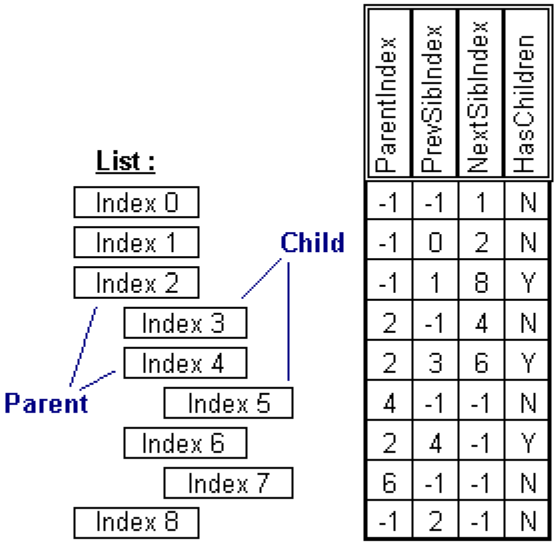
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



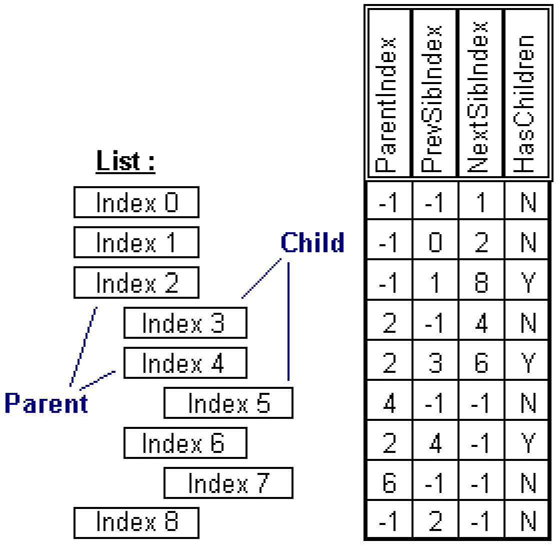
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



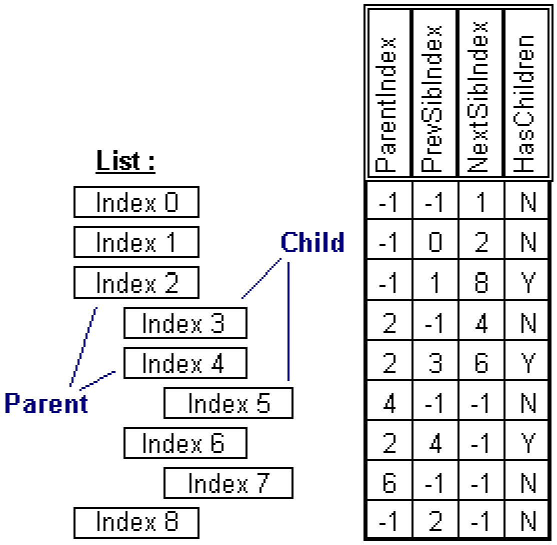
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :
String

Purpose :
Initiate About Box Display

Description :
During design-time, this property will initiate the display of the control's About Box.

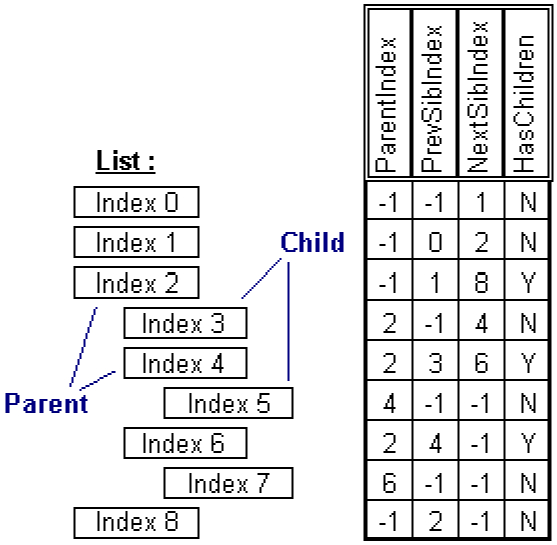
Property Access :
Property Window : Yes
Des Time : Read-Only
Run Time : Read-Only

The AbsPath Property

See also :

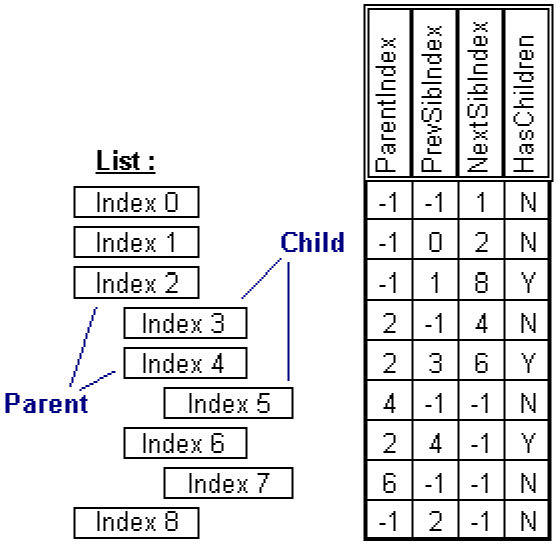
Tree Path Properties
RelPath

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

String, Property Array

Purpose :

an item's path constructed with an item's index

Description :

The AbsPath Property returns a path-like string which identifies an Item in the List. To build this string, the List starts with the specified Item and works its way up through the Parent Items, adding the Index number for each Item as it goes. Indexes listed in the AbsPath Property are separated by periods '.'.

This property is useful to determine if an Item is part of another Item's Ancestry.   Essentially, the AbsPath string is the list of indexes which constitute an Item's Parental Ancestry.

Example :

```
- Getting Values with Code :
Dim S As String
' Set the PathChar to a SemiColon
List.PathChar = ";"
' Add Some Items to the List
List.AddItem "ItemA"
List.AddItem "ItemB"
List.ItemIndent(List.NewIndex) = 1
List.AddItem "ItemC"
List.ItemIndent(List.NewIndex) = 2
List.AddItem "ItemD"
List.ItemIndent(List.NewIndex) = 2
List.AddItem "ItemE"
List.ItemIndent(List.NewIndex) = 1
' The Resulting List would Look Like :
'   ItemA
'      ItemB
'         ItemC
'         ItemD
'      ItemE
' Get RelPath of 'ItemD'
S = List.RelPath(3)
' Path would Look Like :
'   "ItemA;ItemB;ItemD"
' Get AbsPath of 'ItemD'
S = List.AbsPath(3)
' Path would Look Like :
'   "0.1.3"
```

Property Access :

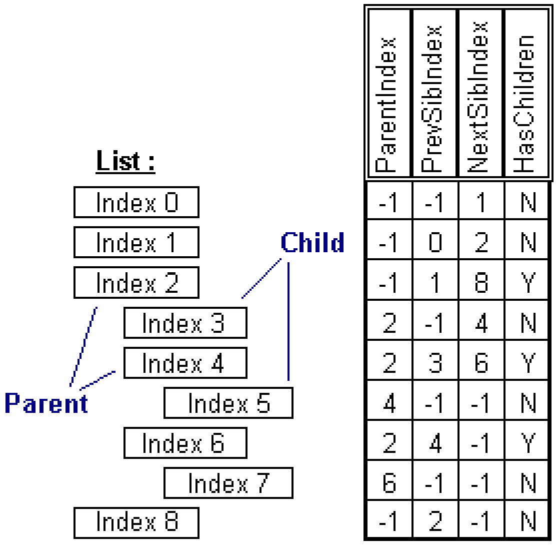
Property Window : No
Des Time : none
Run Time : Read-Only

The BadMove Property

See also :

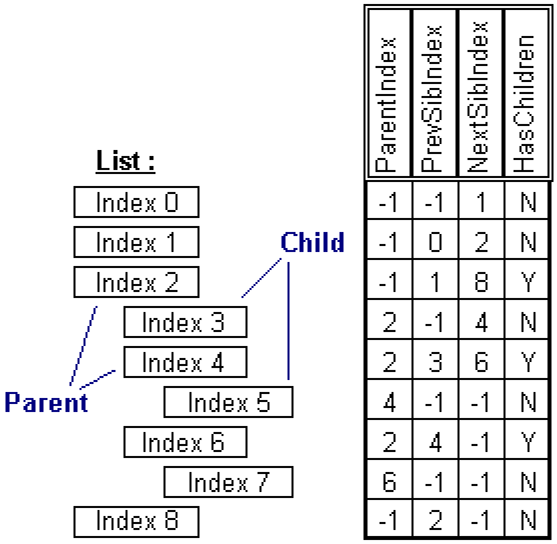
Tree Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree


List :
Index 0
Index 1
Index 2
Index 3
Index 4
Index 5
Index 6
Index 7
Index 8
Child
Parent

ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas

Property Type :

Boolean

Purpose :

indicates if a move was successful

Description :

This property allows you to provide error checking during Tree Traversals and other move operations.   This property works similar to the Visual Basic's Err flag.

The BadMove Property is used to verify that a move operation (through the use of the MoveCurr, MoveToChild, or CurrPath Properties) worked properly or not.   The value of this property is reset anytime one of the move properties is used, or the value may be altered within the program code itself.

Example :

```
- Setting Values with Code :
' Reset the BadMove Error Flag
Tree.BadMove = False
```

```
- Getting Values with Code :
Dim I As Integer
Dim S As String
' Save the CurrPath of the Current Node
S = Tree.CurrPath
' Goto the First Node
Tree.MoveCurr = 1
' Count the Number of Siblings
I = 0
While Not Tree.BadMove
    I = I + 1
    Tree.MoveCurr = 4 ' Get Next Node
Wend
' Reset Current Node
Tree.CurrPath = S
MsgBox "Found " + Str$(I) + " Node(s)"
```

Property Access :

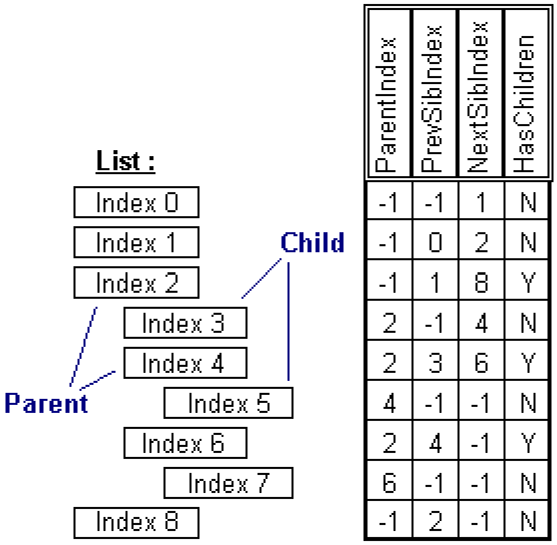
Property Window : No
Des Time : none
Run Time : Read-Write

The BatchMode Property

See also :

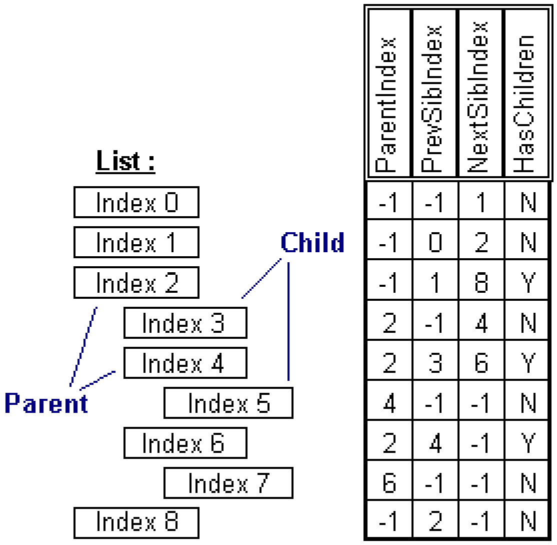
Control Properties

Applies To :



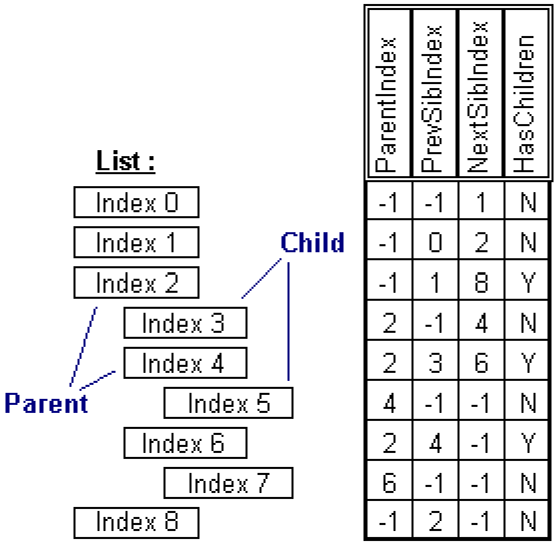
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



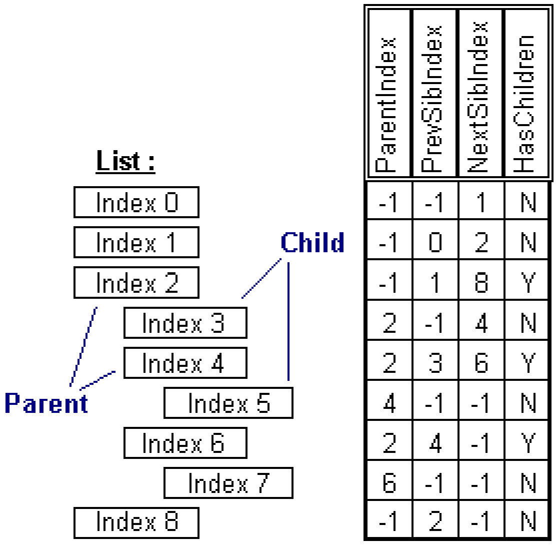
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList



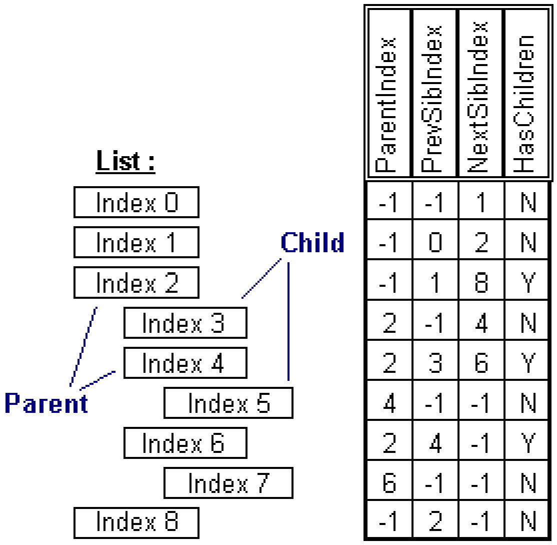
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



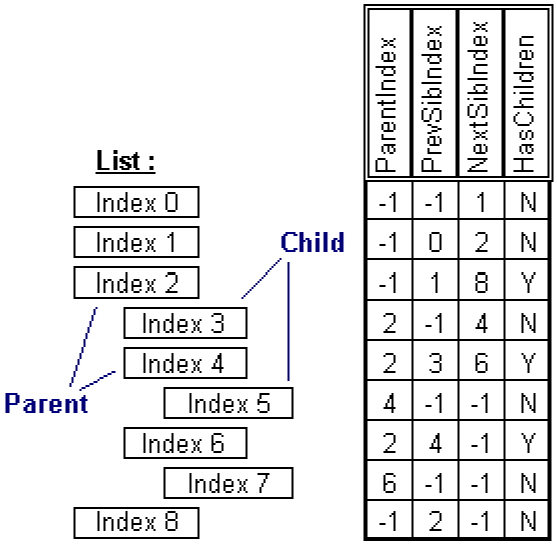
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse



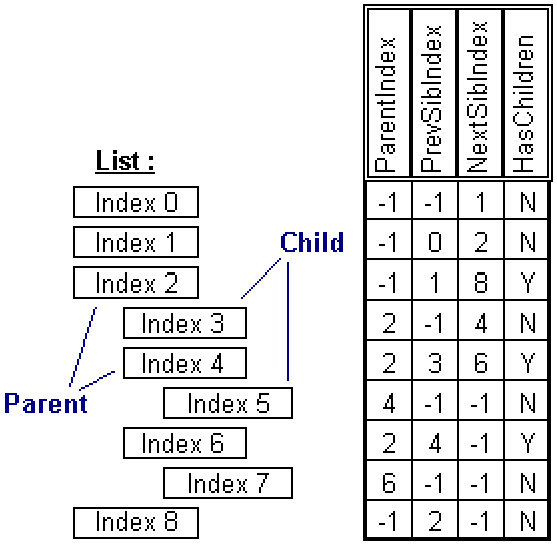
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



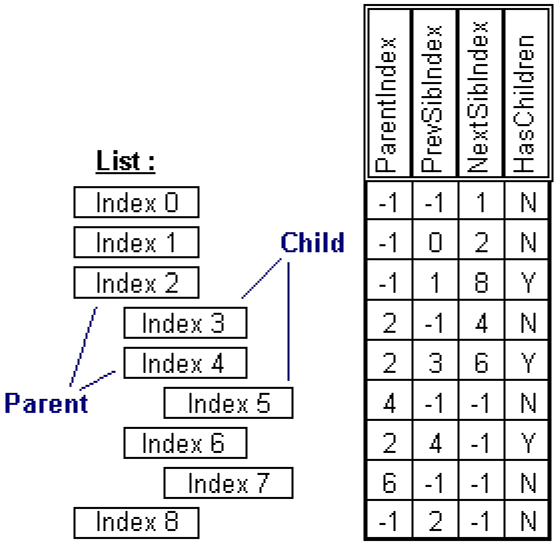
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



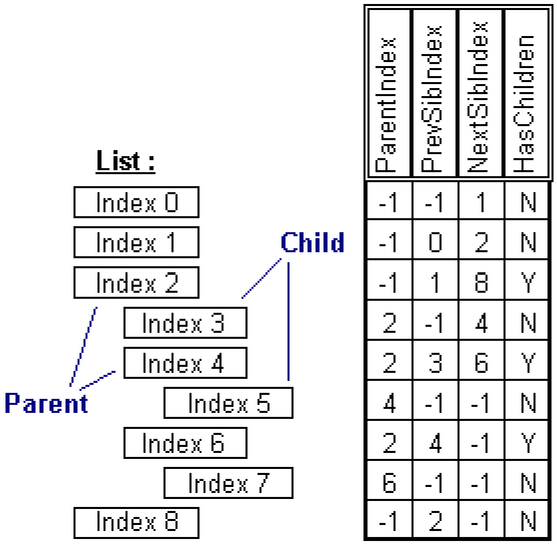
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Boolean

Purpose :

obtain dramatic improvements in speed when performing operations that affect the display

Description :

The BatchMode Property controls the automatic updating of the listbox on the screen when the contents of the List have been changed. This property is usually set to False to allow for automatic screen updates whenever Items are added, deleted, or visually modified; However, setting BatchMode to True will suspend the control's screen updates and will also ignore some memory "housekeeping" chores. The result is that adding, changing, and deleting items in the list is much faster. When this Property is reset to False, the listbox display will immediately be refreshed, and the memory housekeeping will be resumed.

Note: Since the BatchMode Property suspends some of the normal ongoing update process, try not to keep the BatchMode set for long periods of time. If you are setting the BatchMode Property when trying to get a million items from disk into your List, then the BatchMode setting isn't going to make a whole ton of difference anyway.

Example :

```
- Setting Values with Code :
Dim I As Integer
List.BatchMode = True ' Suspend Screen Updates
' Add 1000 Items
For I = 1 To 1000
    List.AddItem ("Item #" + Str$(I))
Next I
List.BatchMode = True ' Reset the Screen Updates

Getting Values with Code :
Dim I As Integer
I = List.BatchMode
```

Property Access :

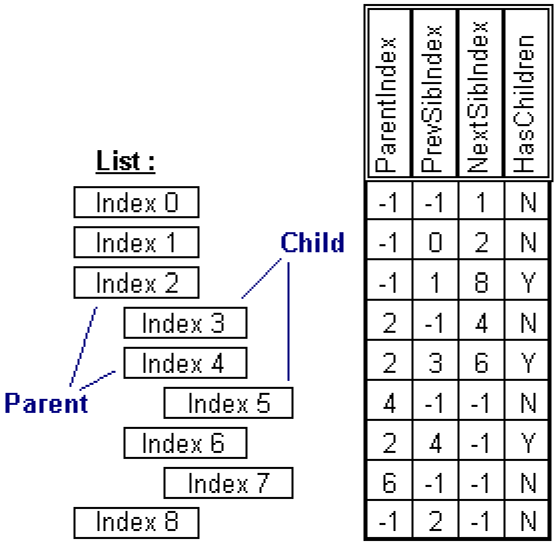
Property Window : No
Des Time : none
Run Time : Read-Write

The bCaseEnforce Property

See also :

Parsing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse

Property Type :

Boolean

Purpose :

used to automatically control case

Description :

The bCaseEnforce Property can be True or False. This value controls whether LParse will enforce the case of parsed Symbols and Keywords when rebuilding the ParseString.
This property is only useful when the bCaseSensitive Property is False. When LParse parses the ParseString and finds a text Symbol or Keyword, having the bCaseEnforce Property set to True will cause the LParse to store the token as it appears in the Symbols or Keywords List.

Example :

```
(See demo code in Trainer LINE PARSER)
- Setting Values with Code :
 LParse.bCaseEnforce = True

- Getting Values with Code :
Dim I As Integer
I = LParse.bCaseCaseEnforce
```

Property Access :

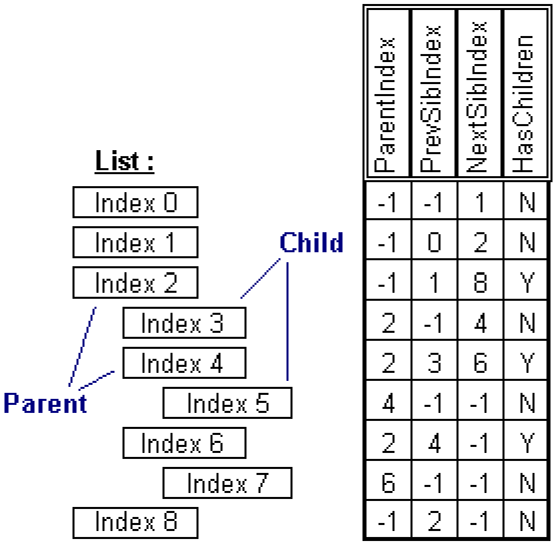
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The bCaseSensitive Property

See also :

Parsing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse

Property Type :

Boolean

Purpose :

control case-sensitivity when parsing

Description :

The bCaseSensitive Property can be True or False. This value controls whether any Symbols or Keywords found in the ParseString are identified with matching case (or regardless of case).
This property is used when you need to identify Symbols and Keywords that are case sensitive.

Example : (See demo code in Trainer LINE PARSER)

```
- Setting Values with Code :
 LParse.bCaseSensitive = True

- Getting Values with Code :
Dim I As Integer
I = LParse.bCaseSensitive
```

Property Access :

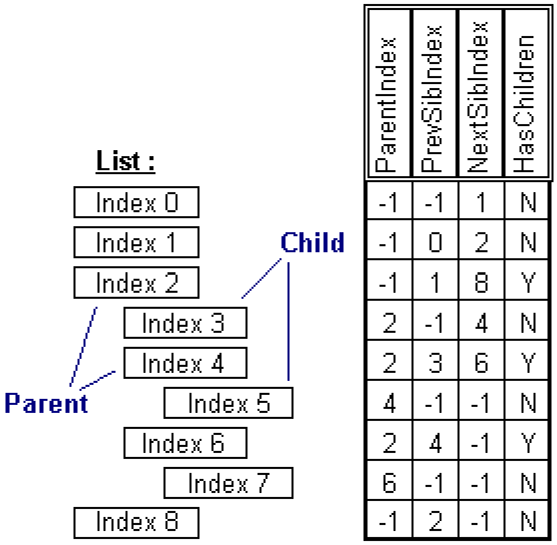
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The bDetectInteger Property

See also :

Parsing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse

Property Type :

Boolean

Purpose :

detect and treat integer numbers as tokens

Description :

This property is useful for finding embedded numerics in your ParseString.
The bDetectInteger Property can be True or False. This value allows LParse to automatically detect numerics (delimited by whitespace and symbols) within the ParseString. When this value is True and LParse finds a digit as the first character of a token, all consecutive digits will be parsed out as a single token and placed in the Parsed String List. This property is similar to, yet differs from, the bDetectReal Property. That is, the bDetectReal Property will read past a decimal point where the bDetectInteger Property will not.

Example : (See demo code in Trainer LINE PARSER)

```
- Setting Values with Code :
 LParse.bDetectInteger = True

- Getting Values with Code :
Dim I As Integer
I = LParse.bDetectInteger
```

Property Access :

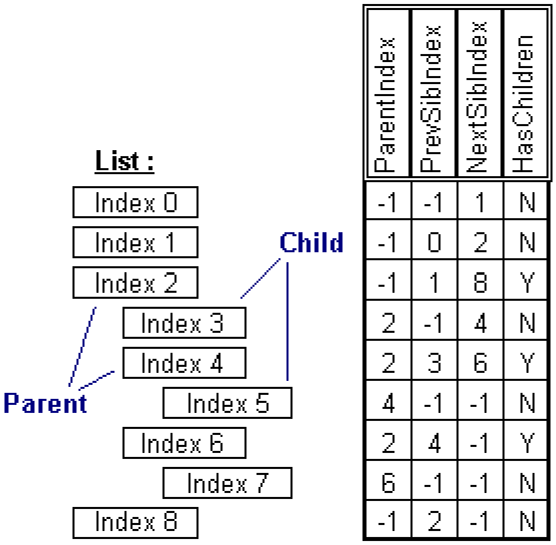
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The bDetectReal Property

See also :

Parsing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse

Property Type :

Boolean

Purpose :

detect and treat real numbers as tokens

Description :

This property is useful for finding embedded numerics in your ParseString.
The bDetectReal Property can be True or False. This value allows LParse to automatically detect numerics (delimited by whitespace and symbols) within the ParseString.   When this value is True and LParse finds a digit (or a decimal followed by a digit) as the first character of a token, all consecutive digits will be parsed out as a single token and placed in the Parsed String List. This property is similar to, yet differs from, the bDetectInteger Property. That is, the bDetectReal Property will read past a decimal point where the bDetectInteger Property will not.

Example :   (See demo code in Trainer LINE PARSER)

```
- Setting Values with Code :
 LParse.bDetectReal = True

- Getting Values with Code :
Dim I As Integer
I = LParse.bDetectReal
```

Property Access :

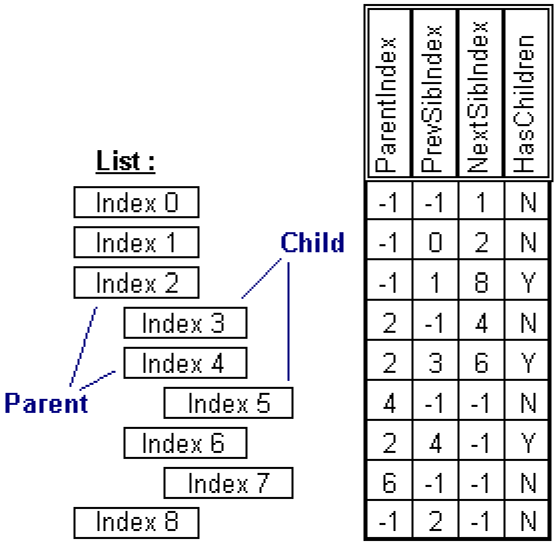
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The bDoubleQuotes Property

See also :

Parsing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse

Property Type :

Boolean

Purpose :

used to treat a quoted string as a single token

Description :

This property is useful when you need to extract quoted, literal strings from within another string
The bDoubleQuotes Property can be True or False. This value controls whether LParse will automatically detect double quoted (") strings. If this value is True and a double quote character is encountered in the ParseString, LParse will get everything up to the next double quote character (or the end of string) and treat that as a single token in the Parsed String List.

Example : (See demo code in Trainer LINE PARSER)

```
- Setting Values with Code :
Dim I As Integer
I = True
LParse.bDoubleQuotes = I

- Getting Values with Code :
Dim I As Integer
I = LParse.bDoubleQuotes
```

Property Access :

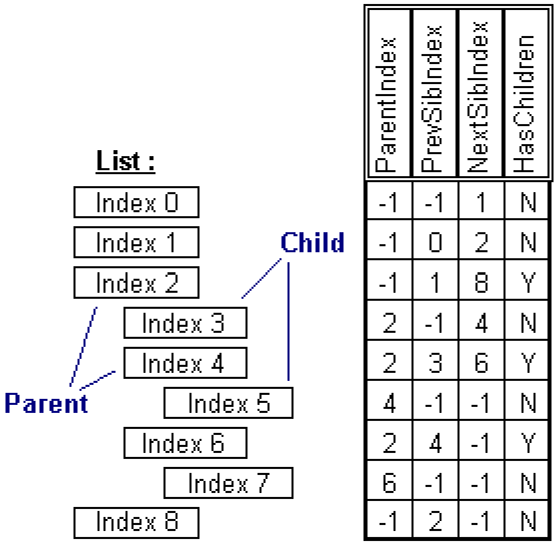
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The BevelDark Property

See also :

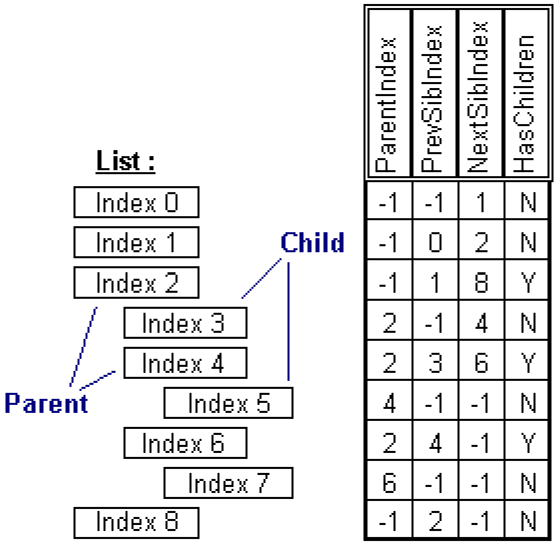
Control Border Properties

Applies To :



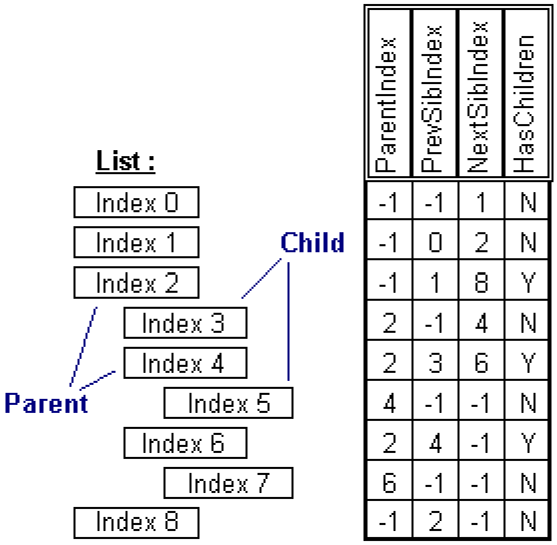
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



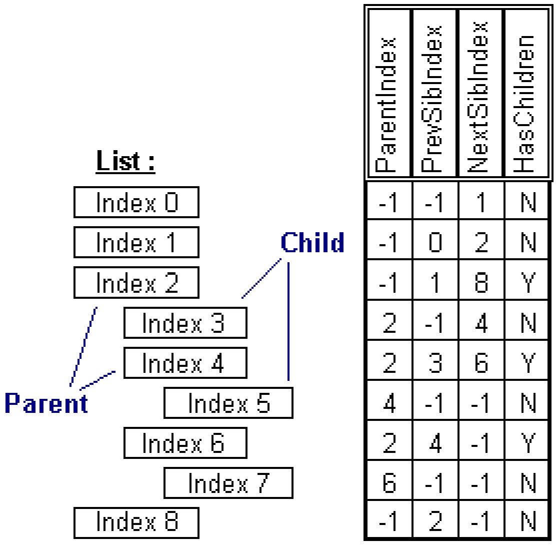
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



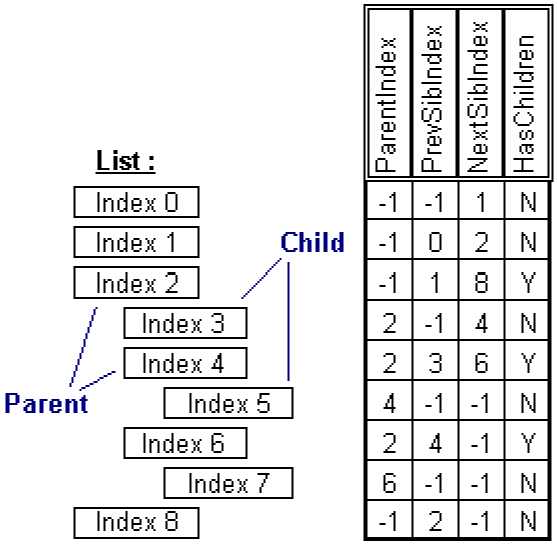
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Long (Color)

Purpose :

Color used for Sunken Bevelling

Description :

This property specifies the color to use when drawing the three-d bevelling.
For the three-d effects, the light source is always assumed to come from the top-left corner of the screen.
Three-d elements which are raised up on the top and left sides are colored with the BevelLight color.   The same color is used for elements which are lowered on the bottom and right sides.
Three-d elements which are lowered on the top and left sides are colored with the BevelDark color.   The same color is used for elements which are raised on the bottom and right sides.
Any width left over after the bevelling will be painted with the BorderColor.

Example :

```
- Setting Values :
' A Crazy Blue Three-D Border
ctrl.BorderColor = QBColor(8)  ' Light Blue
ctrl.BevelLight = QBColor(15)  ' White
ctrl.BevelDark = QBColor(1)  ' Blue
' Set Border Widths and Flags
ctrl.BorderWidth = 5
ctrl.BevelInner = 2
ctrl.BevelOuter = -2
ctrl.ControlFlags = "+BRD_BORDER+BRD_BEVEL"
```

Property Access :

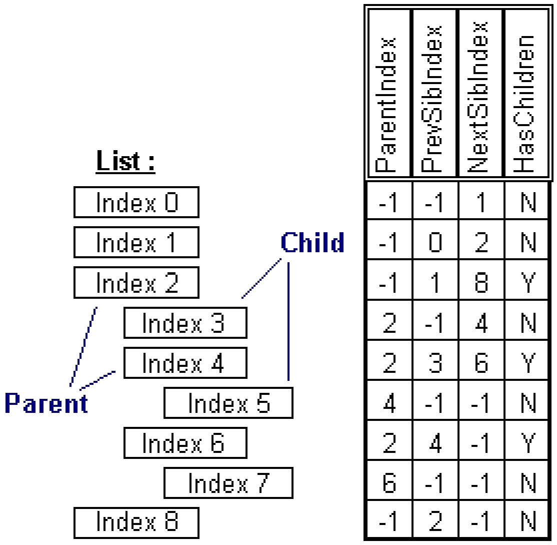
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The BevelInner Property

See also :

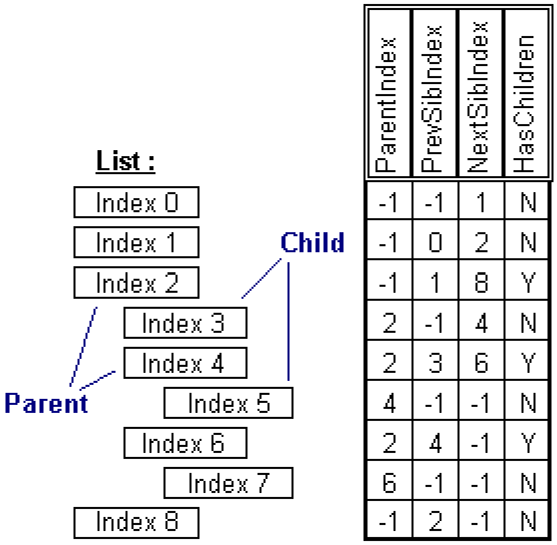
Control Border Properties

Applies To :



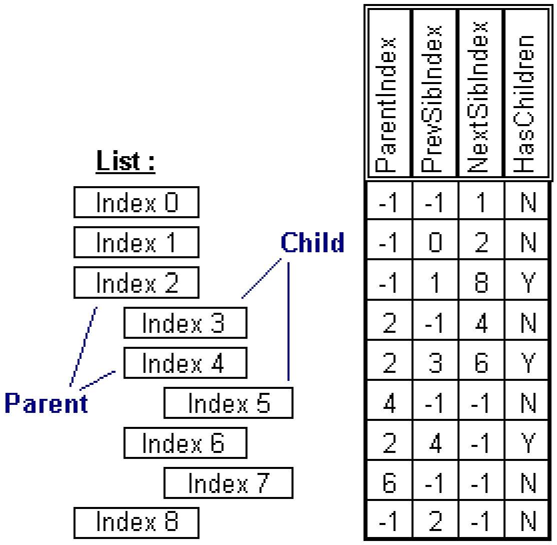
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



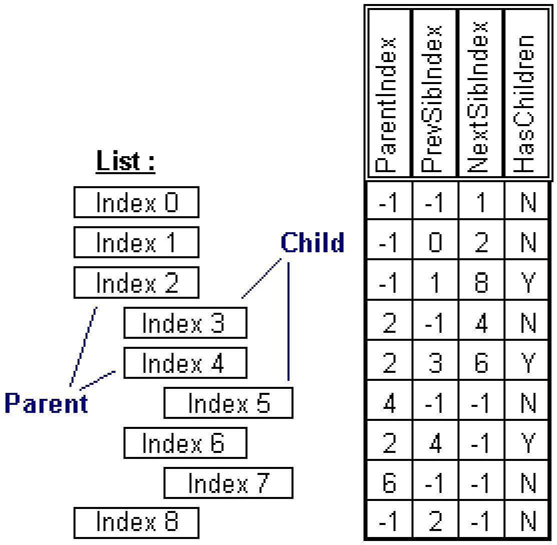
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



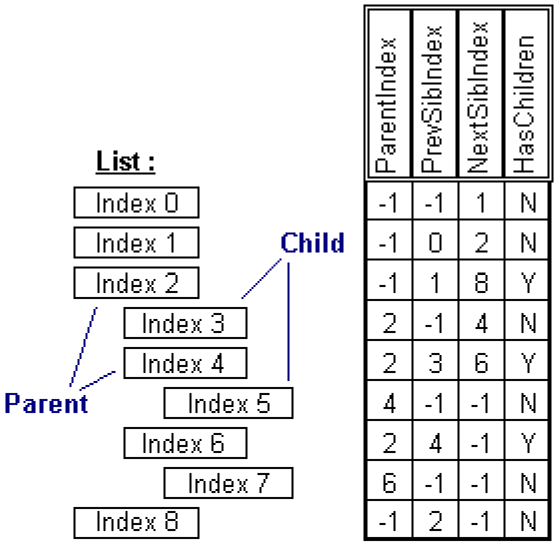
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Integer

Purpose :

Amount to Bevel Inside edge of Border

Description :

The BevelInner Property allows you to specify the width of the bevelling to use on the inside edge of the control's border. A negative value will cause the inside edge of the border as lowered, a positive value displays as rasied.

Note that the BRD_BORDER and BRD_BEVEL flags must be set in the ControlFlags to show the border and bevelling at all.

Example :

```
- Setting Values :
' Set a Small Raised Border around the control
ctrl.BevelInner = 1       ' Inside raised up 1
ctrl.BevelOuter = -1     ' Outside lowered down 1
ctrl.BorderWidth = 3    '  just enough to include the bevelling
                              '   and a pixel for the middle of the border
' Set Flags to Show Border and Bevelling
ctrl.ControlFlags = "+BRD_BORDER+BRD_BEVEL"
```

Property Access :

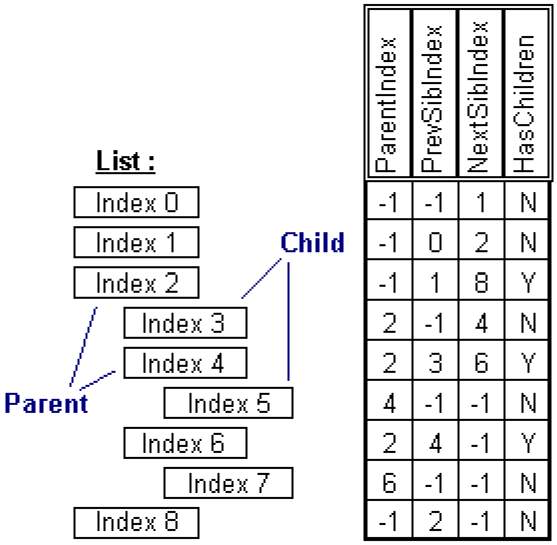
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The BevelLight Property

See also :

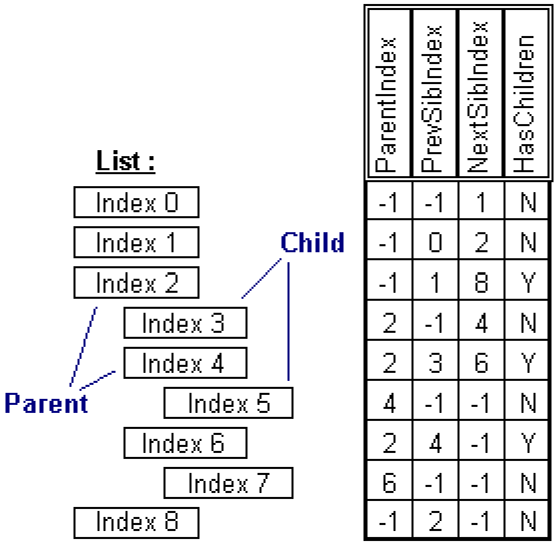
Control Border Properties

Applies To :



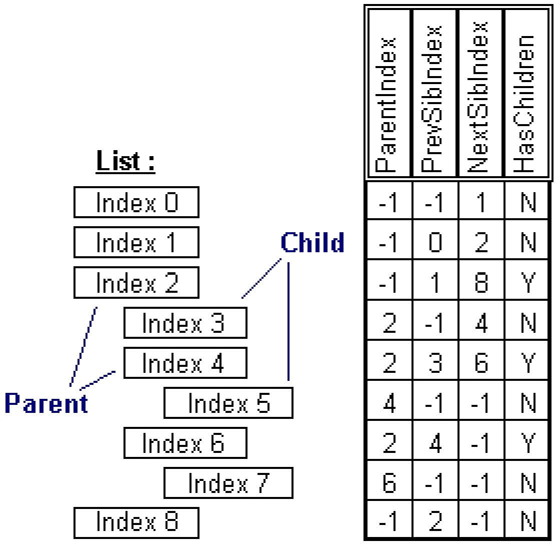
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



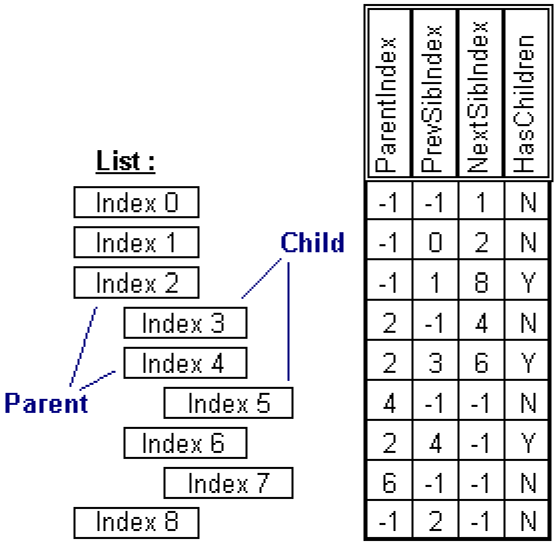
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



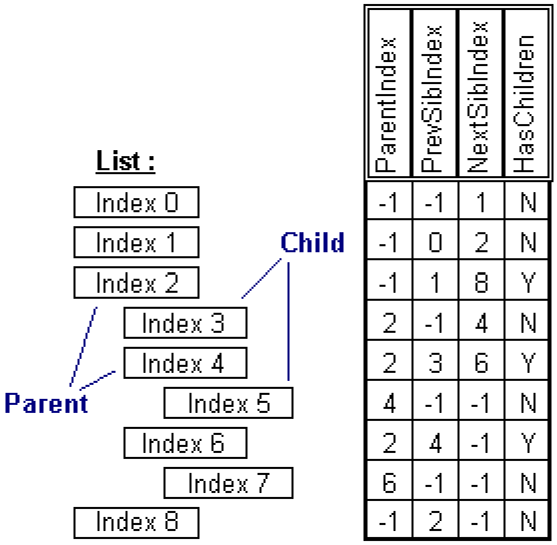
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Long (Color)

Purpose :

Color used for Raised Bevelling

Description :

This property specifies the color to use when drawing the three-d bevelling.
For the three-d effects, the light source is always assumed to come from the top-left corner of the screen.
Three-d elements which are raised up on the top and left sides are colored with the BevelLight color.   The same color is used for elements which are lowered on the bottom and right sides.
Three-d elements which are lowered on the top and left sides are colored with the BevelDark color.   The same color is used for elements which are raised on the bottom and right sides.
Any width left over after the bevelling will be painted with the BorderColor.

Example :

```
- Setting Values :
' A Crazy Blue Three-D Border
ctrl.BorderColor = QBColor(8) ' Light Blue
ctrl.BevelLight = QBColor(15) ' White
ctrl.BevelDark = QBColor(1) ' Blue
' Set Border Widths and Flags
ctrl.BorderWidth = 5
ctrl.BevelInner = 2
ctrl.BevelOuter = -2
ctrl.ControlFlags = "+BRD_BORDER+BRD_BEVEL"
```

Property Access :

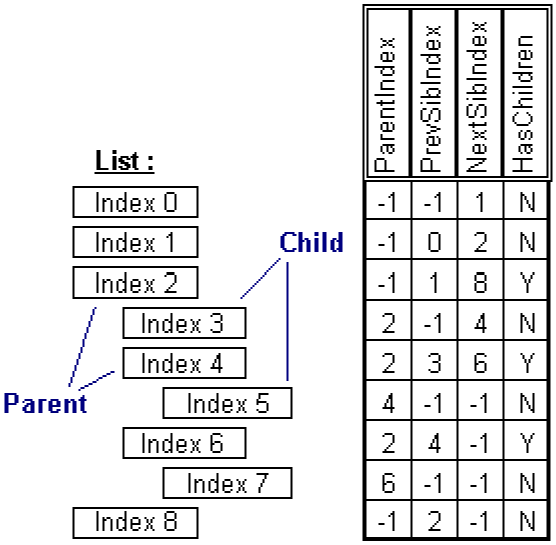
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The BevelOuter Property

See also :

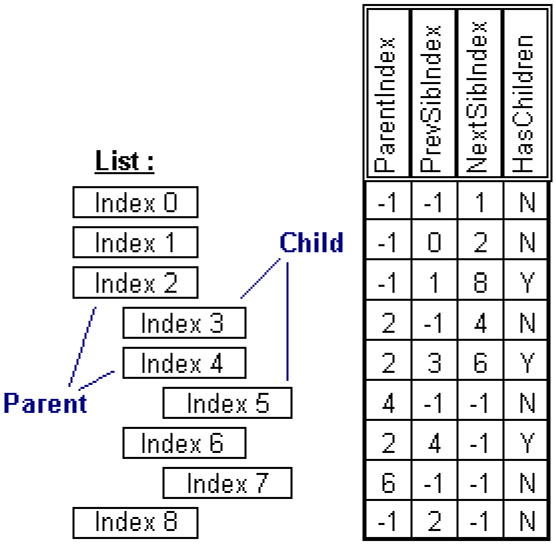
Control Border Properties

Applies To :



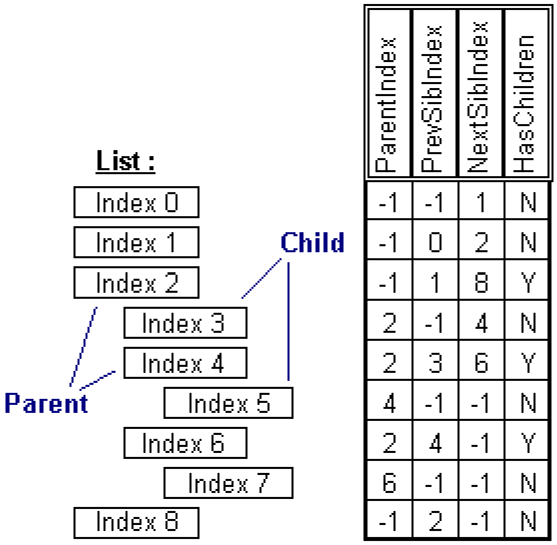
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



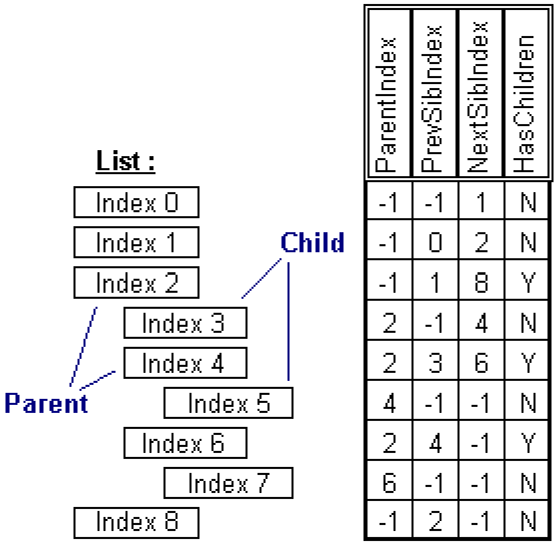
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



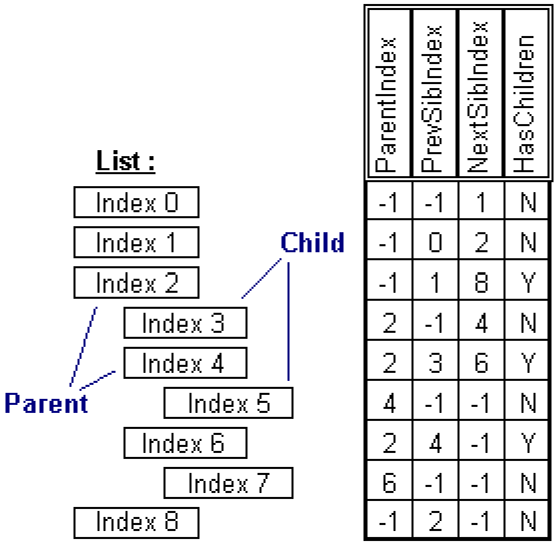
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Integer

Purpose :

Amount to Bevel Outside edge of Border

Description :

The BevelOuter Property allows you to specify the width of the bevelling to use on the outside edge of the control's border. A negative value will cause the outside edge of the border as lowered, a positive value displays as rasied.

Note that the BRD_BORDER and BRD_BEVEL flags must be set in the ControlFlags to show the border and bevelling at all.

Example :

```
- Setting Values :
' Set a Small Raised Border around the control
ctrl.BevelInner = 1      ' Inside raised up 1
ctrl.BevelOuter = -1     ' Outside lowered down 1
ctrl.BorderWidth = 3     ' just enough to include the bevelling
                         ' and a pixel for the middle of the border
' Set Flags to Show Border and Bevelling
ctrl.ControlFlags = "+BRD_BORDER+BRD_BEVEL"
```

Property Access :

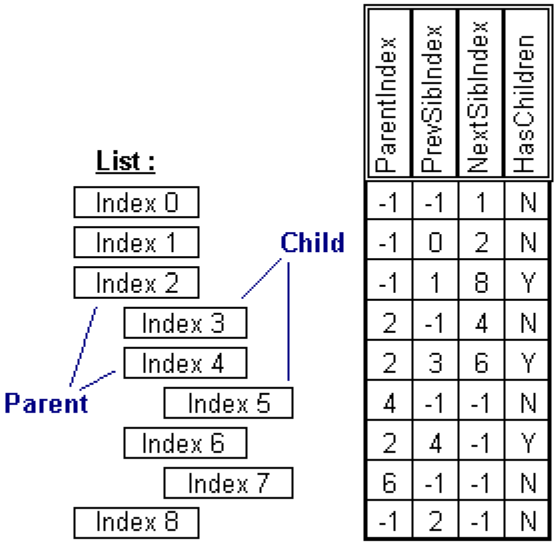
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The BookMark Property

See also :

Data Browsing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree

Property Type :

String, Property Array

Purpose :

position the underlying Data Control to a specific record

Description :

The BookMark Property returns a BookMark string which is compatible with the Data Control identified by the BrowseSource Property. Each Record that is Browsed and Listed in the List has its own BookMark string.

Note: This property is useful for recalling and positioning the underlying Data Control to a specific record.

Example :

```
see example in BrowseSource Property
```

Property Access :

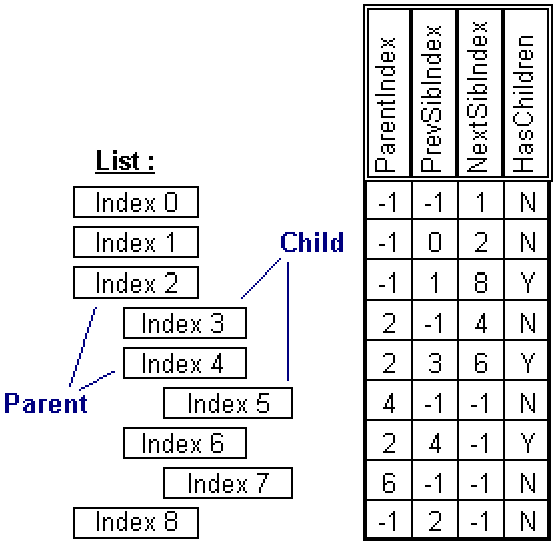
Property Window : No
Des Time : none
Run Time : Read-Only

The BorderColor Property

See also :

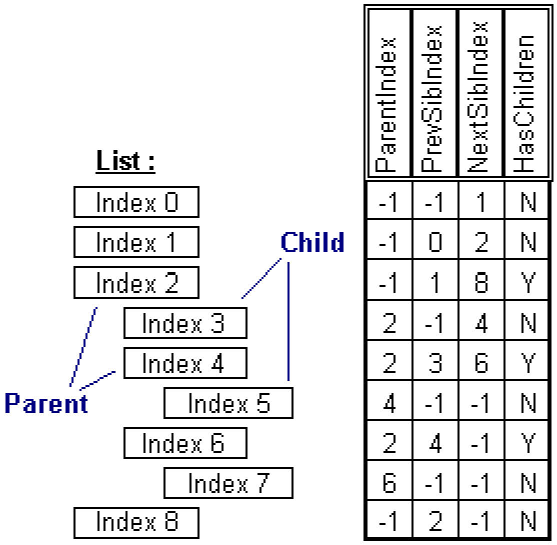
Control Border Properties

Applies To :



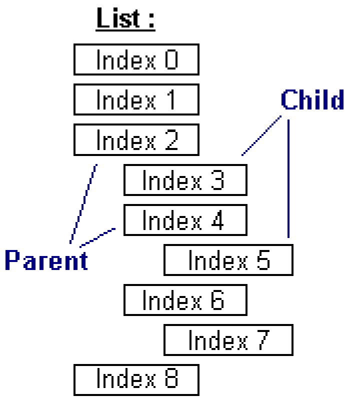
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



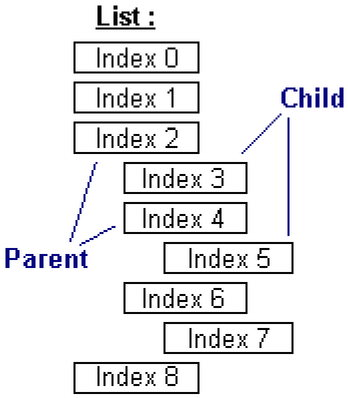
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



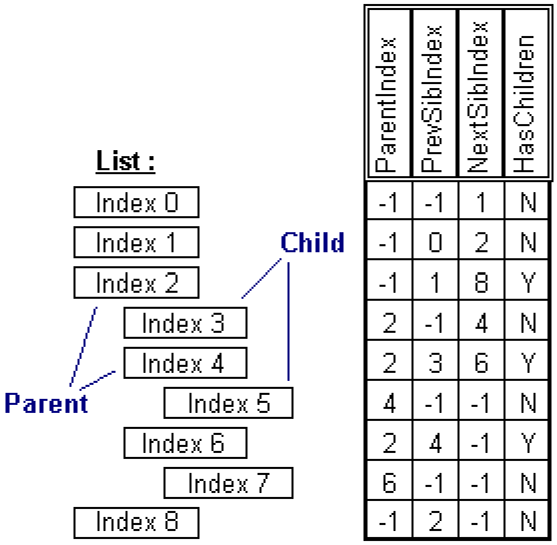
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :
Long (Color)

Purpose :
Color of Border

Description :
This property specifies the color to use when drawing the control's border.
For the three-d effects, the light source is always assumed to come from the top-left corner of the screen.
Three-d elements which are raised up on the top and left sides are colored with the BevelLight color. The same color is used for elements which are lowered on the bottom and right sides.
Three-d elements which are lowered on the top and left sides are colored with the BevelDark color. The same color is used for elements which are raised on the bottom and right sides.
Any width left over after the bevelling will be painted with the BorderColor.

Example :
```
- Setting Values :
' A Crazy Blue Three-D Border
ctrl.BorderColor = QBColor(8)  ' Light Blue
ctrl.BevelLight = QBColor(15)  ' White
ctrl.BevelDark = QBColor(1)  ' Blue
' Set Border Widths and Flags
ctrl.BorderWidth = 5
ctrl.BevelInner = 2
ctrl.BevelOuter = -2
ctrl.ControlFlags = "+BRD_BORDER+BRD_BEVEL"
```

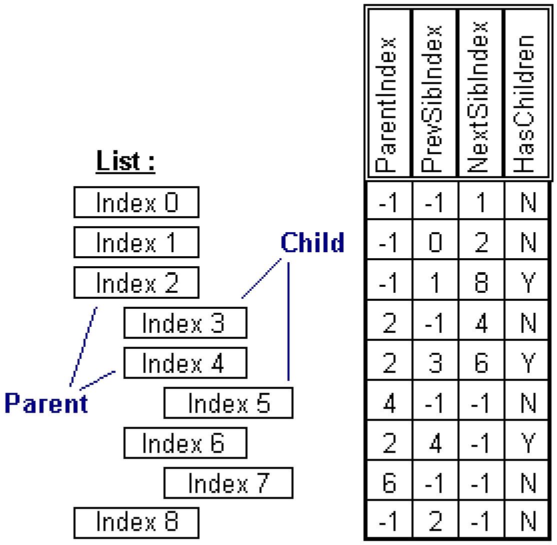
Property Access :
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The BorderWidth Property

See also :

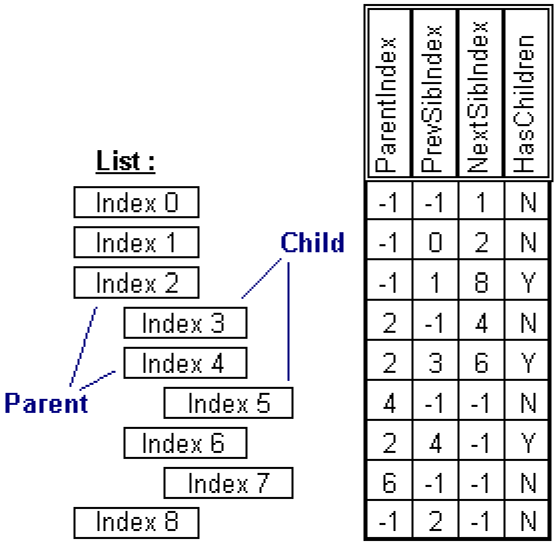
Control Border Properties

Applies To :



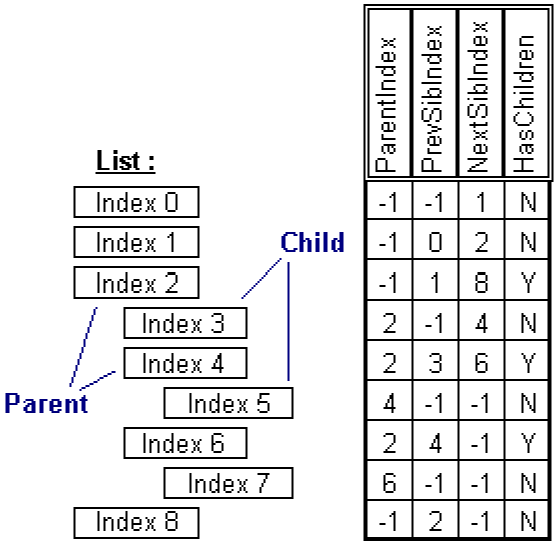
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



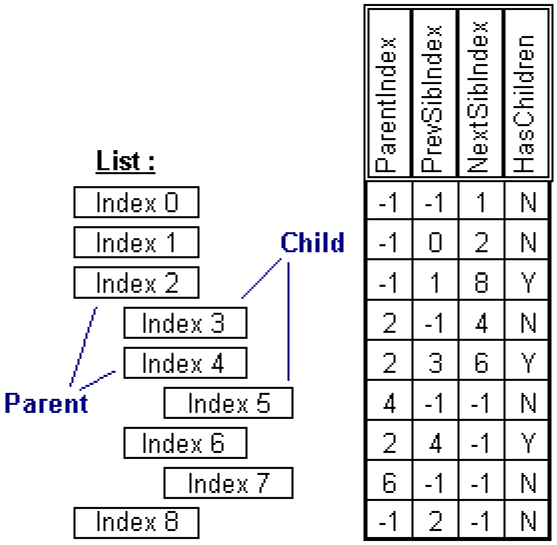
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



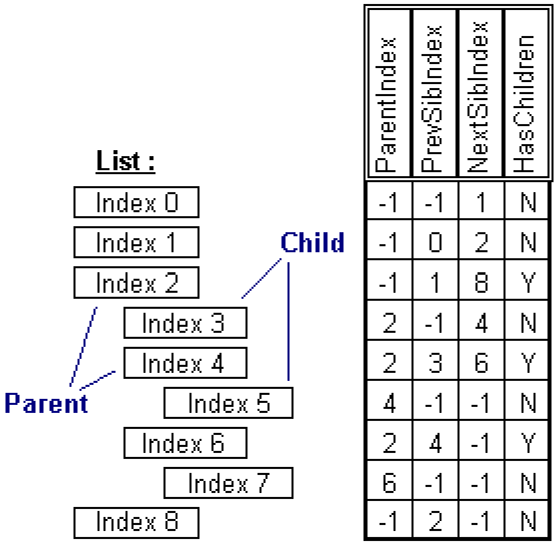
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :
Integer

Purpose :
Total Width of Border and Bevelling

Description :
The BorderWidth Property allows you to specify the width of the bevelled border surrounding your control. This width includes the bevelling widths specified by BevelInner and BevelOuter.

Note that the BRD_BORDER flag must be set in the ControlFlags to show the border at all.

Example :
```
- Setting Values :
' Set a Small Raised Border around the control
ctrl.BevelInner = 1      ' Inside raised up 1
ctrl.BevelOuter = -1     ' Outside lowered down 1
ctrl.BorderWidth = 3     ' just enough to include the bevelling
                         ' and a pixel for the middle of the border
' Set Flags to Show Border and Bevelling
ctrl.ControlFlags = "+BRD_BORDER+BRD_BEVEL"
```

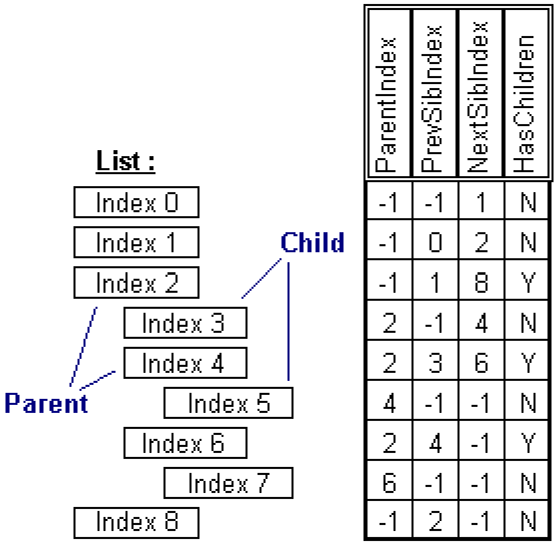
Property Access :
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The BottomIndex Property

See also :

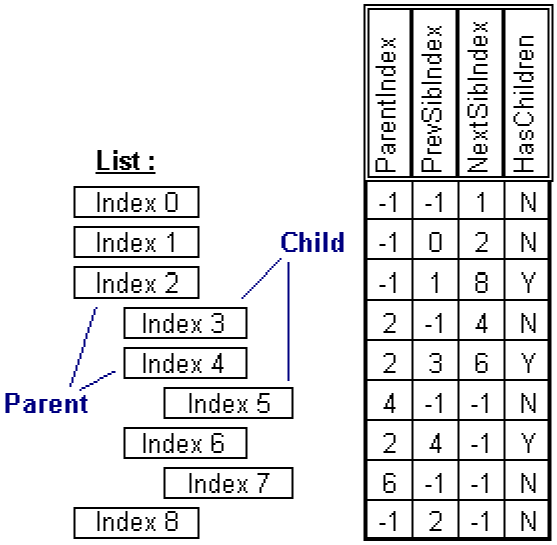
List Data Properties

Applies To :



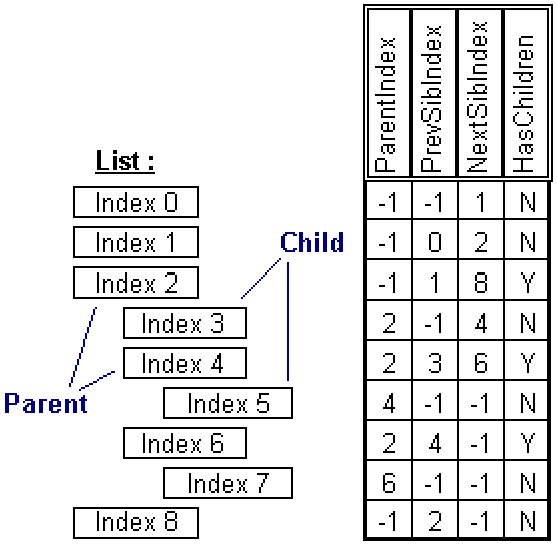
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



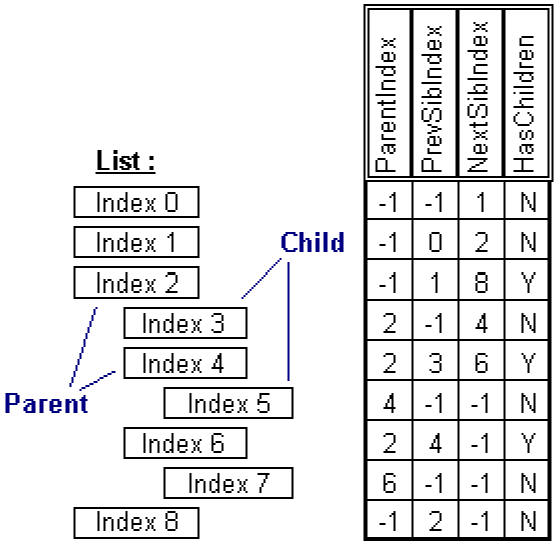
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Long

Purpose :

Last Visible Item at Bottom of List Display

Description :

The TopIndex Property indicates the Index of the Item which is the topmost visible Item in the listbox. This value will automatically change when the listbox is being scrolled.

Note: This property is useful to programmatically scroll the listbox up and down.

Example :

```
- Setting Values with Code :
' Scrolls Top of List
List.BottomIndex =  0
' Scroll to Currently Selected Item
List.BottomIndex = List.ListIndex
' Scroll to End of List
List.BottomIndex = (List.ListCount - 1)
```

```
- Getting Values with Code :
Dim I As Integer
I = List.BottomIndex
```

Property Access :

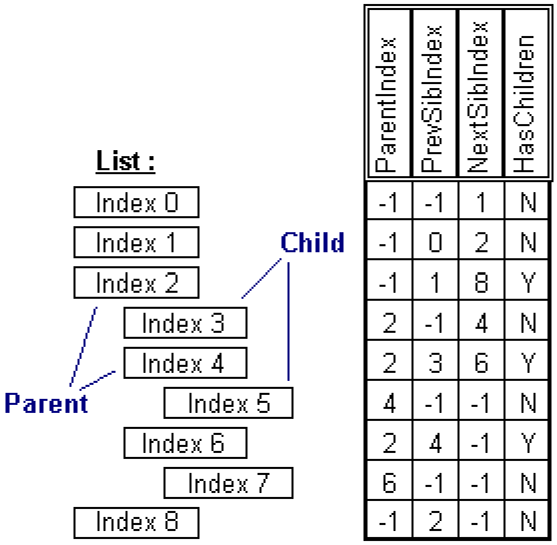
Property Window : No
Des Time : none
Run Time : Read-Write

The BrowseFields Property

See also :

Data Browsing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree

Property Type :

String

Purpose :

display specific Fields from a Data Control's RecordSource Property

Description :

The BrowseFields Property determines what Fields will be displayed within the List while Browsing a Data Control. FieldNames are listed in the BrowseFields string and are separated by a semicolon ';'. Any Field listed must also appear in the RecordSource of the underlying Data Control. The CTree Formatting Codes may also be applied to individual FieldNames.

This property allows you to display specific Fields from a Data Control's RecordSource Property. Each Record in the Data Control will appear as an Item in the List, and will be formatted according to the Formatting Codes specified in the BrowseFields Property.

Example :

```
see example in BrowseSource Property
```

Property Access :

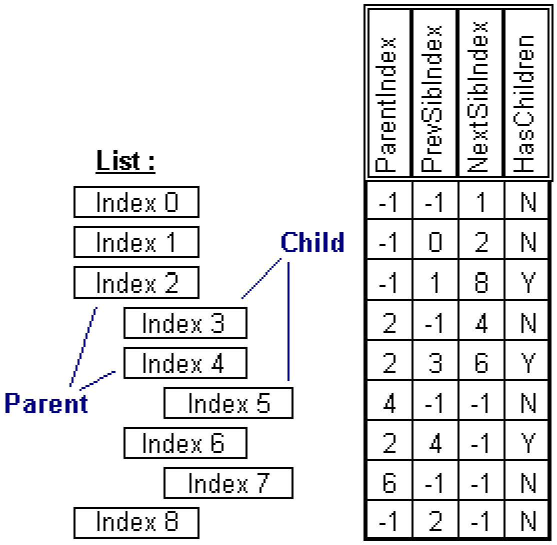
Property Window : No
Des Time : none
Run Time : Read-Write

The BrowseRefresh Property

See also :

Data Browsing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree

Property Type :

Boolean

Purpose :

refresh the List contents for data browsing

Description :

The BrowseRefresh Property signals the List to begin Browsing the Data Control specified in the BrowseSource Property, and Display the Field Values for each record found according to the contents of the BrowseFields Property. To initiate the Browsing Process, simply assign any value other than zero to the BrowseRefresh Property.

This property allows you to control when the List refreshes its contents through the Browsing of a Data Control.

Example :

```
see example in BrowseSource Property
```

Property Access :

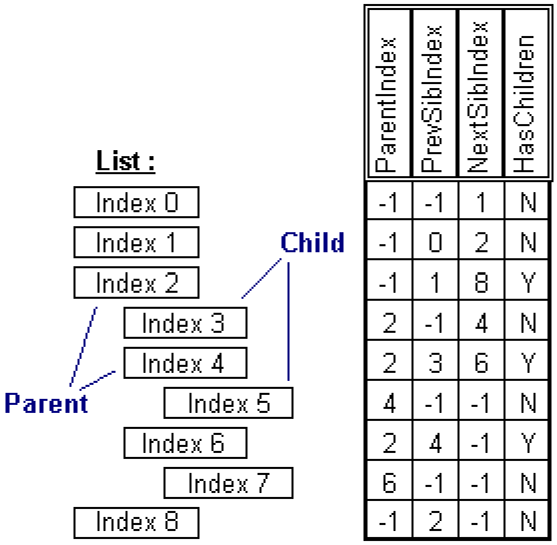
Property Window : No
Des Time : none
Run Time : Write-Only

The BrowseSource Property

See also :

Data Browsing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree

Property Type :

String

Purpose :

identifies the Name of the Data Control to be browsed for data

Description :

This property can be used to switch Source Data Controls for Browsing. This property can also be changed at Run-Time, allowing you to have more control and reusability with the browsing process.

The BrowseSource Property identifies the Name of the Data Control to be browsed for data.

Example :

```
- Setting Values with Code :
Dim I As Integer
' SetUp the BrowseSource
List.BrowseSource = 'dataEmployee'
' SetUp the BrowseFields
'    EmployeeName is Bolded
'    Address is Blue and starts at Col 20
List.BrowseFields = '\BEmpName;\CFBlu\TC20\Address'
' Do the Browse
List.BrowseRefresh = True
' Position the Data Control to the Last Record
I = (List.ListCount - 1)
dataEmployee.RecordSet.BookMark = List.BookMark(I)

- Getting Values with Code :
Dim S As String
S = List.BrowseSource
```

Property Access :

Property Window : Yes
Des Time : Read-Write, Saved

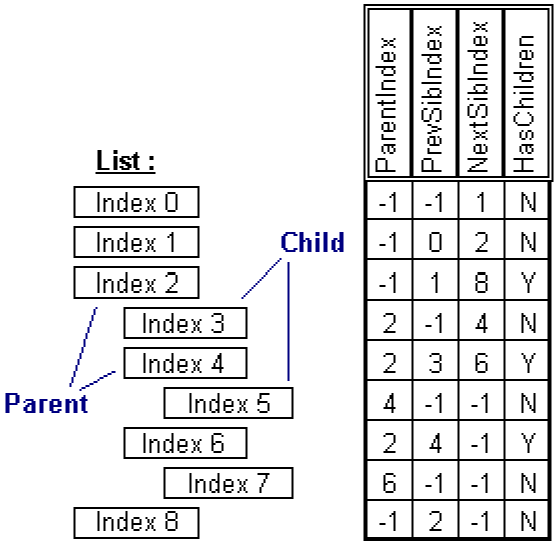
Run Time : Read-Write

The bSingleQuotes Property

See also :

Parsing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse

Property Type :

Boolean

Purpose :

used to treat a quoted string as a single token

Description :

This property is useful when you need to extract quoted, literal strings from within another string
The bSingleQuotes Property can be True or False. This value controls whether LParse will automatically detect single quoted (') strings. If this value is True and a single quote character is encountered in the ParseString, LParse will get everything up to the next single quote character (or the end of string) and treat that as a single token in the Parsed String List.

Example : (See demo code in Trainer LINE PARSER)

```
- Setting Values with Code :
 LParse.bSingleQuotes = True

- Getting Values with Code :
Dim I As Integer
I = LParse.bSingleQuotes
```

Property Access :

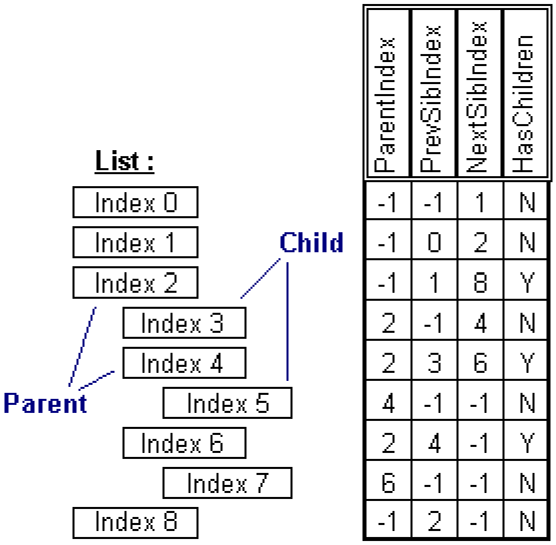
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The ChildCount Property

See also :

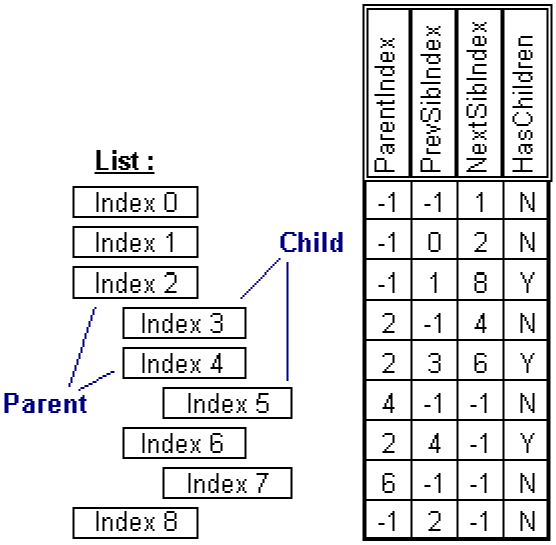
Tree Properties

Applies To :



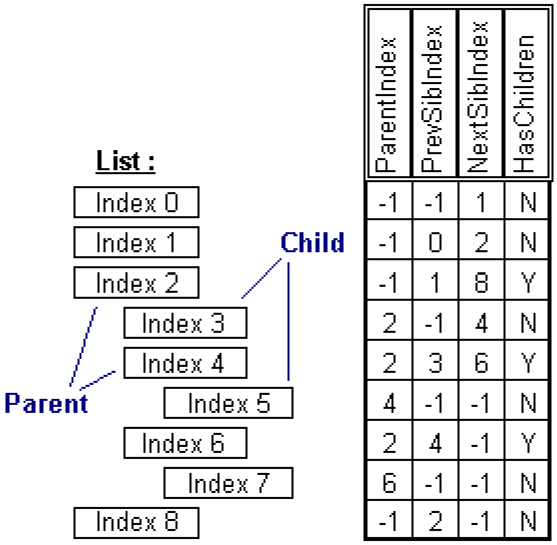
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Long, Property Array

Purpose :

number of children for a specific item

Description :

The ChildCount Property returns the Number of Children found for a particular Child Node. This property may be more accurately described as getting the number of GrandChildren Nodes under a particular Child Node. This property is a Property Array, and therefore, requires the Index of the Child Node in question to appear within the parenthesis.
This property can be useful to determine if a Child Node has more Children under it before moving to the Child.

Example :

```
- Getting Values with Code :
Dim I As Integer
Dim cChild As Integer
Dim cGChild As Integer
' Init Counters
cChild = Tree.ListCount
cGChild = 0
' Count All Grand Children
For I = 0 To (cCount - 1)
    ' Add Number of Children Under Child #I
    cGChild = cGChild + Tree.ChildCount(I)
Next I
MsgBox "Found " + Str$(cChild) + " Children, and " + Str$(cGChild) + " GrandChildren."
```

Property Access :

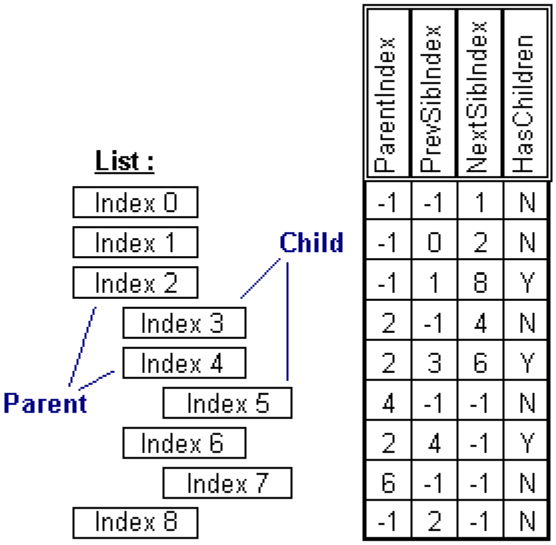
Property Window : No
Des Time : none
Run Time : Read-Only

The ChildrenVisible Property

See also :

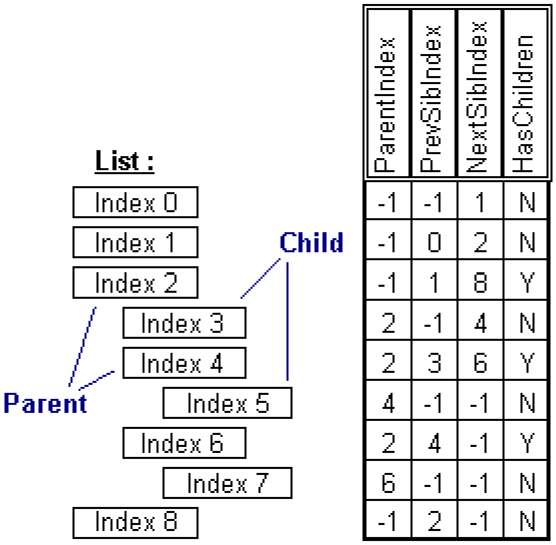
Tree Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Boolean, Property Array

Purpose :

determines the visibility for children of a certain item

Description :

This property lets you programatically expand and collapse a parent item.   When expanding a parent item, only the first level of children will be displayed.   When collapsing an item, all children will be hidden and their ItemVisible( ) properties will be set to false.

Example :

```
- Getting Values with Code :
```

```
Dim I as Long
For I = 0 to (lst.ListCount-1)
        If lst.ChildCount(I) > 0 then
                 Expand Parent Item to Show Children
                lst.ChildrenVisible(I) = true
        End if
Next I
```

Property Access :

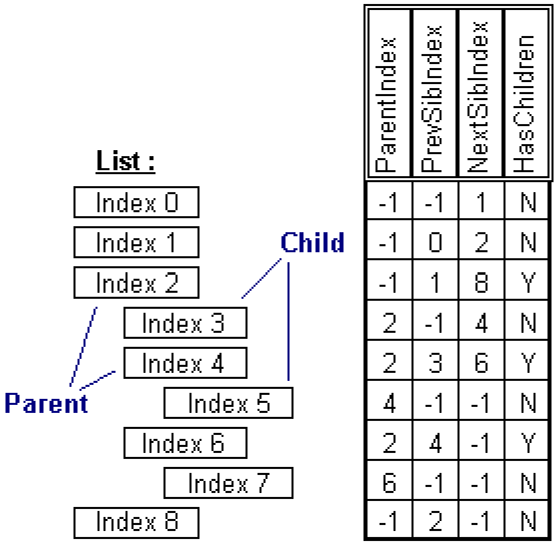
Property Window : No
Des Time : none
Run Time : Read-Write

The ClonePropsGet Property

See also :

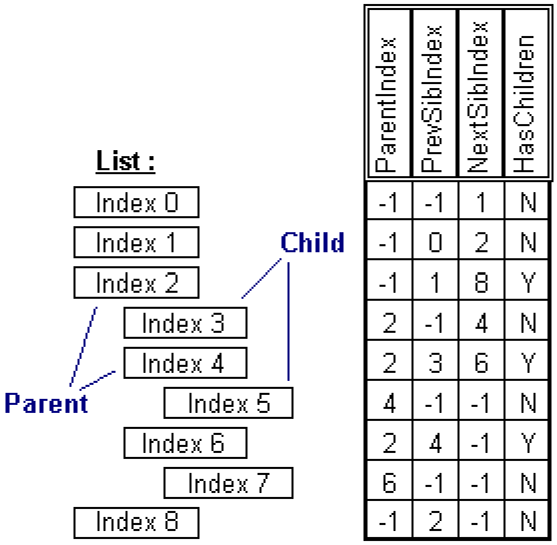
Other Properties

Applies To :



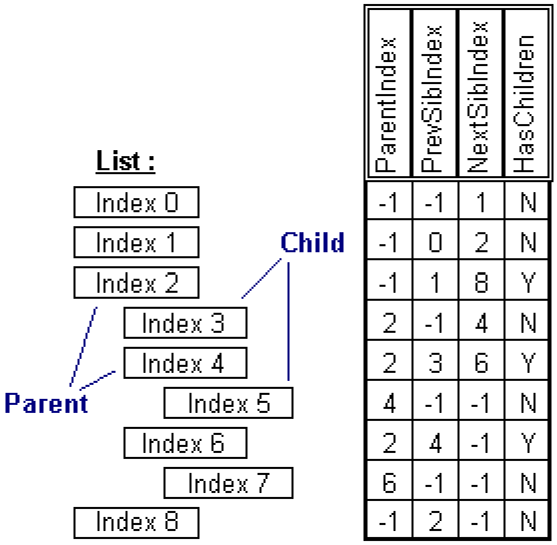
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



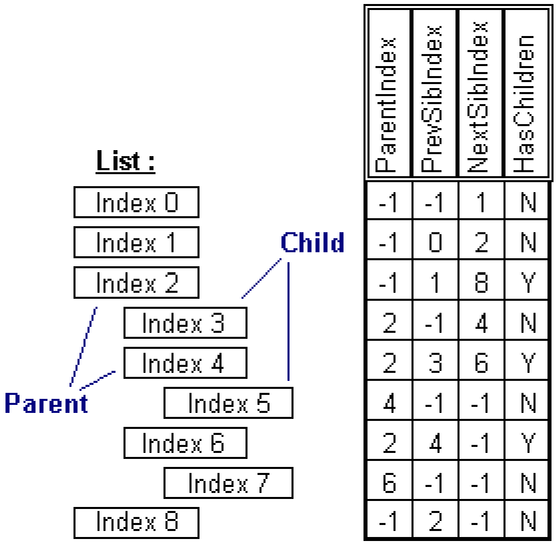
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList



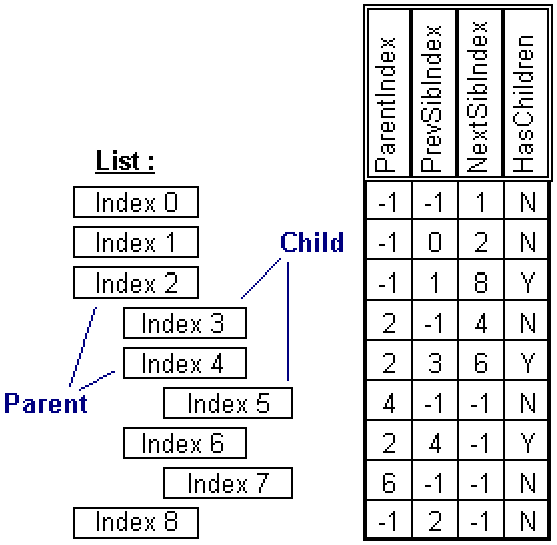
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



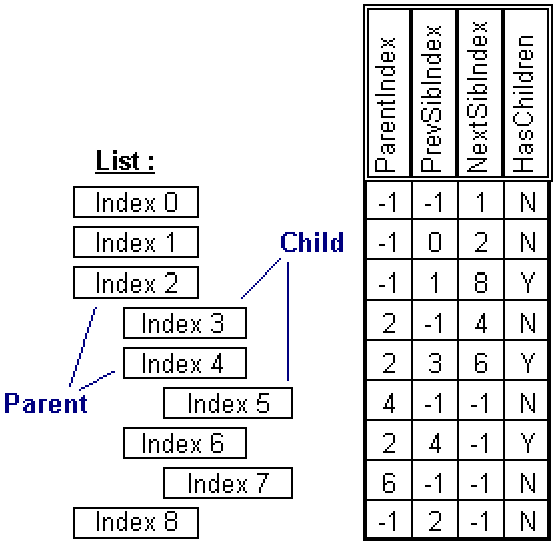
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



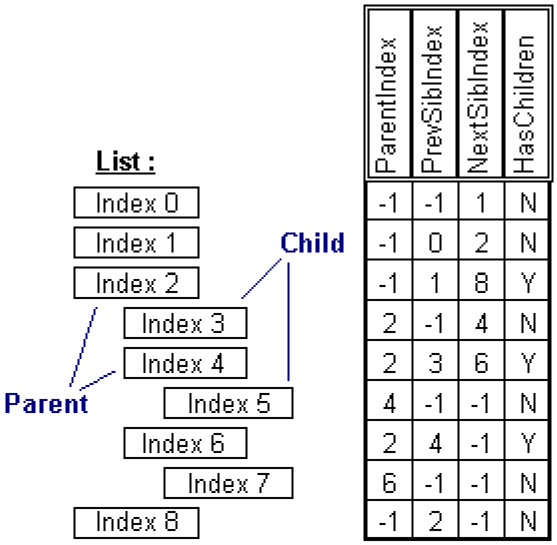
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Integer

Purpose :

Used to Transfer Property Settings

Description :

The ClonePropsGet and ClonePropsSet properties are used to transfer property settings and data between controls of the same control class.   These properties were originally designed for use by the configuration program but have been included to allow you to write and easily implement your own configuration programs, or just to copy data between controls.
When this property is set with the hWnd of another control (of the same class), the control will copy all propertiy settings and items from the control specified by hWnd.

Example :

```
- Setting Values with Code :
' Copy props and items from List2 to List1
List1.ClonePropsGet = List2.hWnd
```

Property Access :

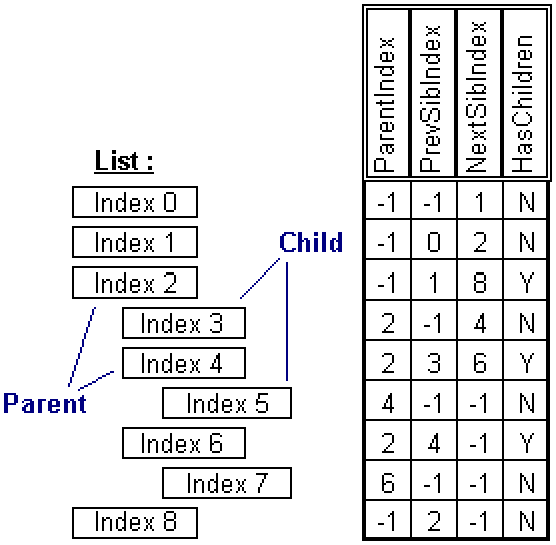
Property Window : No
Des Time : Write-Only
Run Time : Write-Only

The ClonePropsSet Property

See also :

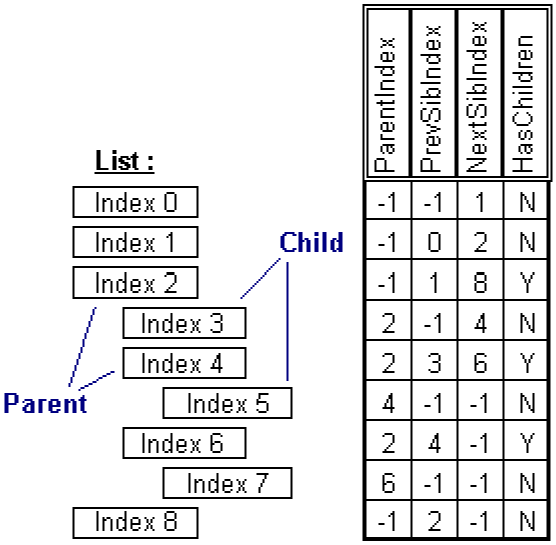
Other Properties

Applies To :



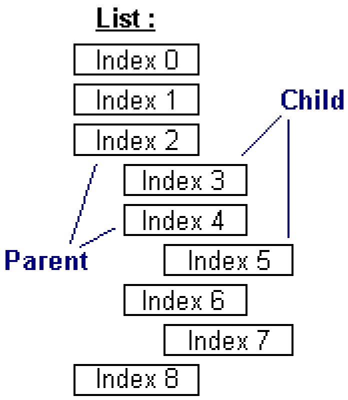
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



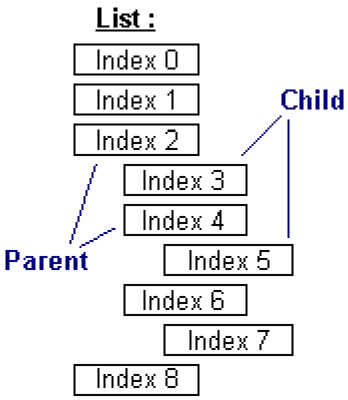
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList



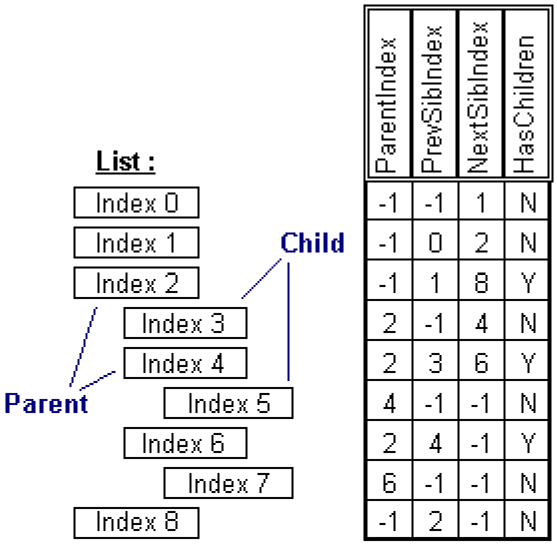
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



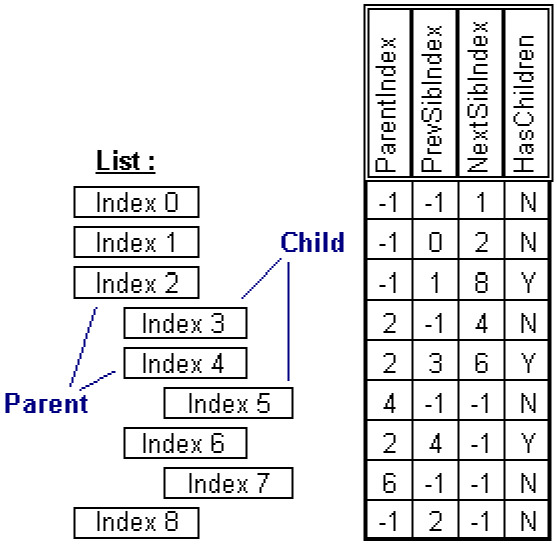
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



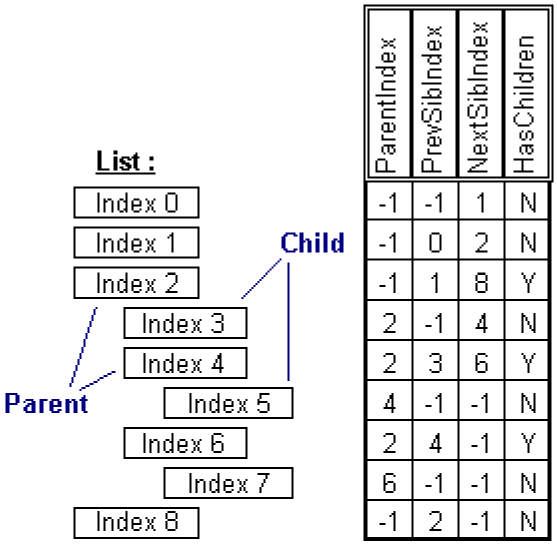
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Integer

Purpose :

Used to Transfer Property Settings

Description :

The ClonePropsGet and ClonePropsSet properties are used to transfer property settings and data between controls of the same control class.   These properties were originally designed for use by the configuration program but have been included to allow you to write and easily implement your own configuration programs, or just to copy data between controls.
When this property is set with the hWnd of another control (of the same class), the control will apply all propertiy settings and items to the control specified by hWnd.

Example :

```
- Setting Values with Code :
' Copy props and items from List1 to List2
List1.ClonePropsSet = List2.hWnd
```

Property Access :

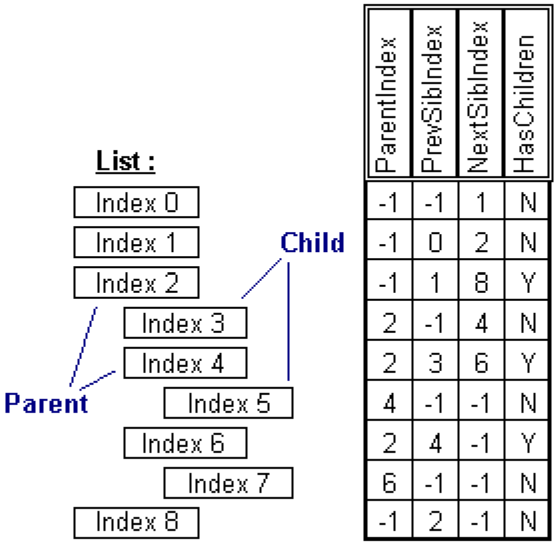
Property Window : No
Des Time : Write-Only
Run Time : Write-Only

The ColumnSpacing Property

See also :

List Display Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas

Property Type :

Integer

Purpose :

spacing to use between columns of items

Description :

This property determines the amount of spacing to put between columns of items.   This value is applied to left of a column of items, in addition to the ItemSpacing property.

Example :

```
- Setting Values with Code :
' Pixels between columns to 25
Dim I as integer
I = 25
lst.ColumnSpacing = I
```

Property Access :

Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

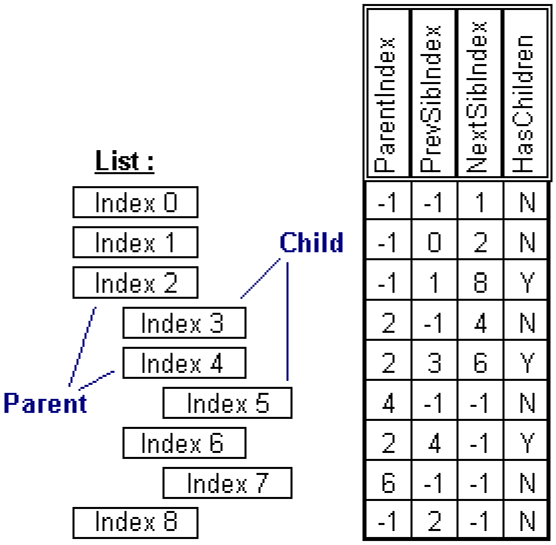
The ControlFlags Property

See also :

Control Properties

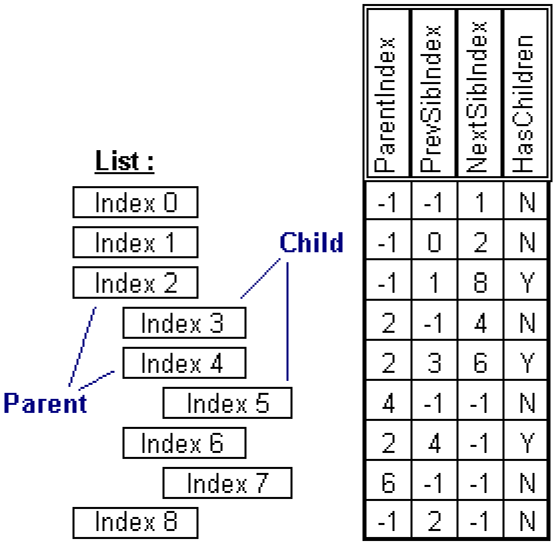
Control Flags String and Long Values

Applies To :



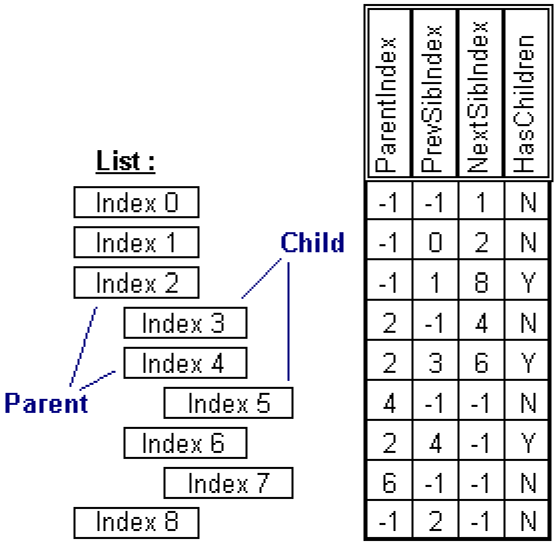
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



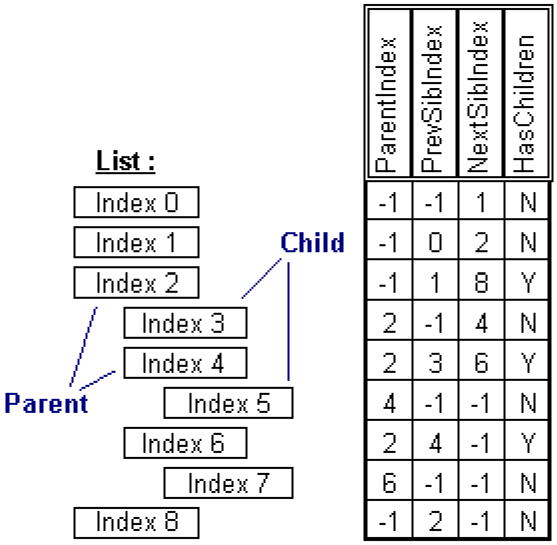
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



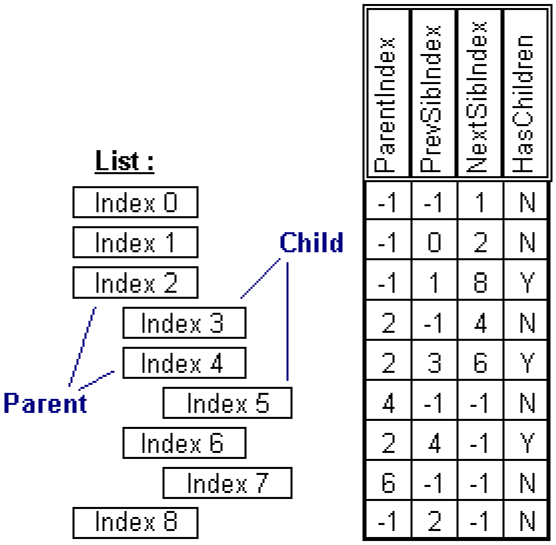
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :
String

Purpose :
set special flags for control display and behaviour

Description :
This property allows you to dynamically alter the display and behaviour of your control easily and efficiently. Using flags (either in string or numeric form) you are able to change the following aspects of control operation : Border Display, Vertical and Horizontal Scrolling, display of a Focus Rectangle, Item Selection, Item Expand/Collapse, and Item Layout. Note that some of these flags only have meaning when applied to certain controls. The TSLabel, for example, has not all to do with selecting multiple items in a listbox.

This property accepts and returns a string value; However, the string values are translated to and from long integer values internally. When using Basic, you can use either strings or longs when manipulating the contents of this property, the values will be converted appropriately. Note that when using long integers with this property, it is very much like using the flags in the MsgBox function.

Setting the value of this property is a little different than when setting most other properties. The most powerful way to set the ControlFlags, is to build a string flags to set or unset. Flags must be seperated by a plus '+' or a minus '-'. When preceeded by a plus, the flag will by set within the control. Conversely, a minus preceeding a flag means that flag will be turned off within the control. When using the string method, The new flags will be added to the current flags to arrive at new settings. This is cool because you can customize certain aspects of the control, and make sure of some of the other settiings at any time. If you use the long integer method, the current flags will be totally replaced by the new flags. The same effect can be forced with the string method by omitting the first plus or minus.

Example :

```
- Setting Values :
Dim s As String
' Keep current settings in control but turn on the control's
' shadow and turn off the scrollbar display
s = "+BRD_SHADOW-VSC_SHOW-HSC_SHOW"
ctrl.ControlFlags = s
```

```
- Getting Values :
Dim L As Long
' Get the Control Flags
L = ctrl.ControlFlags
' Check to see if the Shadow is set
If ((L And BRD_SHADOW) = BRD_SHADOW) Then
    MsgBox "The Shadow is Set!"
End If
```

Property Access :

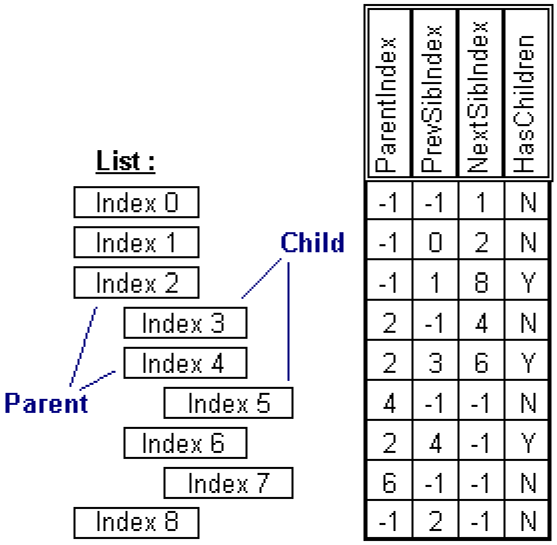
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The ControlVersion Property

See also :

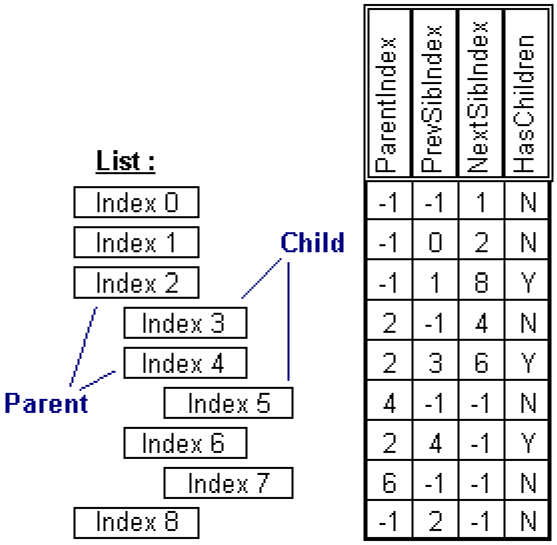
Other Properties

Applies To :



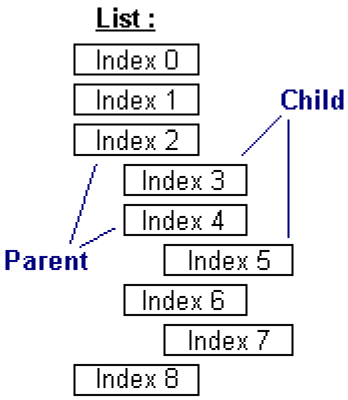
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



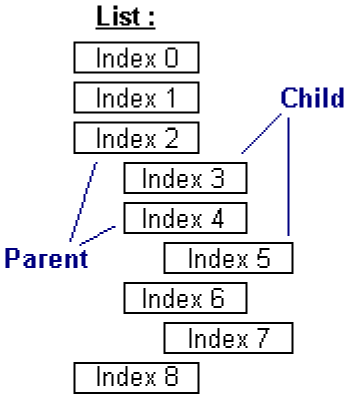
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



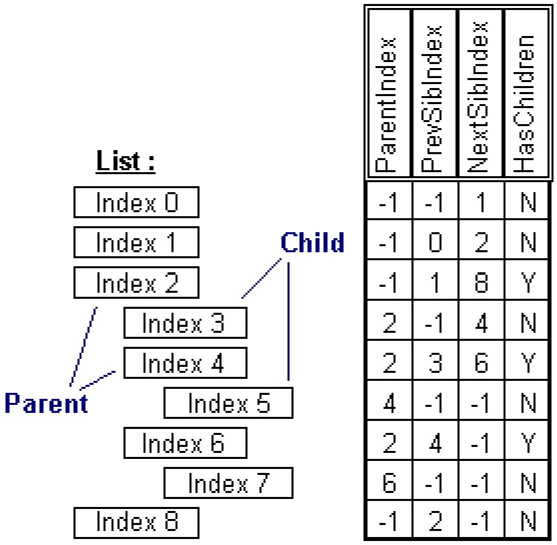
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

String

Purpose :

Displays the Loaded Version of the Control

Description :

During design-time, this property displays the loaded version of the control.

Property Access :

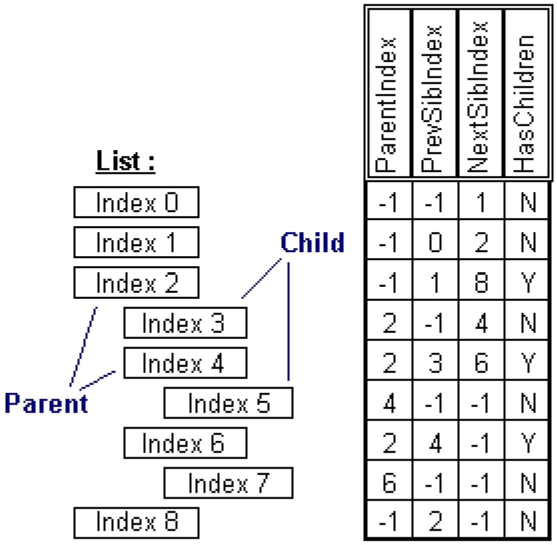
Property Window : Yes
Des Time : Read-Only
Run Time : Read-Only

The CurrDepth Property

See also :

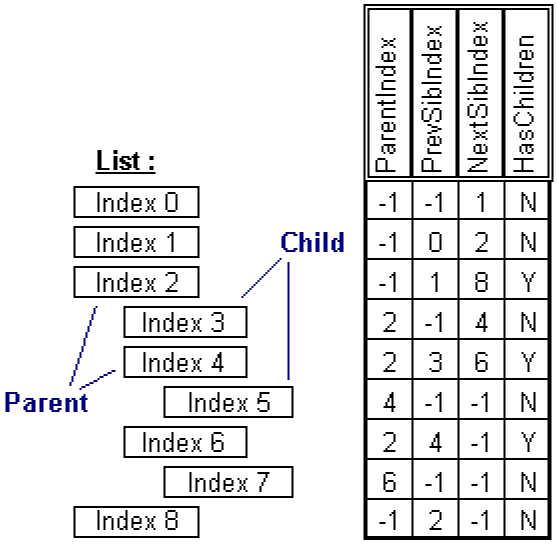
Tree Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas

Property Type :

Integer

Purpose :

the depth of the current node

Description :

The CurrDepth Property returns the number of levels deep the Current Node is. The Root Node's CurrDepth is always 0. This property is useful to give you additional information about the Current Node.

Example :

```
- Getting Values with Code :
Dim S As String
' An Alternate way of Getting to the Root
' Keep Moving to Parent Until you Hit Depth 0
While Tree.CurrDepth > 0
    ' Build a Path from ExtraStrings
    S = Tree.CurrExtraString + "\" + S
    ' Move To Parent
    Tree.MoveCurr = 5
Wend
```

Property Access :

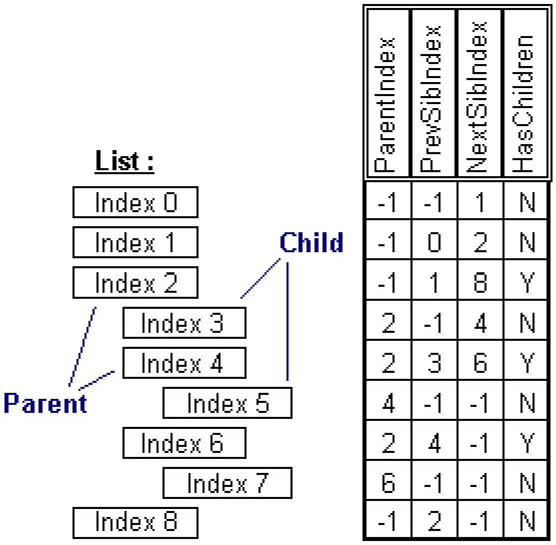
Property Window : No
Des Time : none
Run Time : Read-Only

The CurrExtraString Property

See also :

Item Data Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree

Property Type :

String

Purpose :

secondary string storage

Description :

The CurrExtraString Property is the extra string data associated with the Current Node. This property is the counterpart to the listbox's ItemString Property Array, and has been included to maintain a consistency of properties with the other controls in this package. You may set the CurrExtraString Property of a Node at Run-Time and, by virtue of moving to that node's parent, access that same string later through the ItemString Property.

This property provides an additional storage medium for the Tree. While the most useful way to interact with the Tree is through the ItemText, ItemData, and ItemString Properties (pertaining to the Current List of Children), it is also at times useful to know what the Current Node's ItemText, ItemData and ItemString values are.

Example :

```
- Setting Values with Code :
Dim S As String
' Add a some extra text to a Child Node
S = 'This is some extra text."
Tree.CurrExtraString = S
' Move to the Parent Node
Tree.MoveCurr = 5
' Search for S and get the Index
Tree.SearchType = 2
Tree.SearchString = S
MsgBox (S + ' was found at : ' + Str$(Tree.FoundIndex))
```

```
- Getting Values with Code :
Dim S As String
' Add a Child Node to the Tree with some extra text
Tree.AddItem 'This is a Child Node"
Tree.ItemString( Tree.NewIndex ) = 'This is some extra text."
' Move to That Node and get that same extra text
Tree.MoveToChild = Tree.NewIndex
S = Tree.CurrExtraString
```

Property Access :

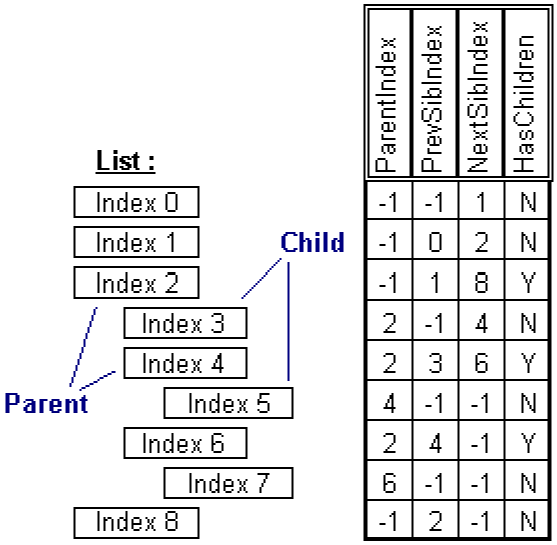
Property Window : No
Des Time : none
Run Time : Read-Write

The CurrItemData Property

See also :

Item Data Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree

Property Type :

Long

Purpose :

main numerical storage

Description :

The CurrItemData Property is the Long Integer data associated with the Current Node. This property is the counterpart to the listbox's ItemData Property Array, and will generally be the Tree's main form of numerical data storage. You may set the CurrItemData Property of a Node at Run-Time and, by virtue of moving to that node's parent, access that same number later through the ItemData Property.

This property provides the primary numerical storage medium for the Tree. While the most useful way to interact with the Tree is through the ItemText, ItemData, and ItemString Properties (pertaining to the Current List of Children), it is also at times useful to know what the Current Node's ItemText, ItemData and ItemString values are.

Example :

```
Dim L As Long
' Get the Number of Current Children Nodes
L = Tree.ListCount
' Store it in the ItemData
Tree.CurrItemData = L
Getting Values with Code :
Dim L As Long
L = Tree.CurrItemData
```

Property Access :

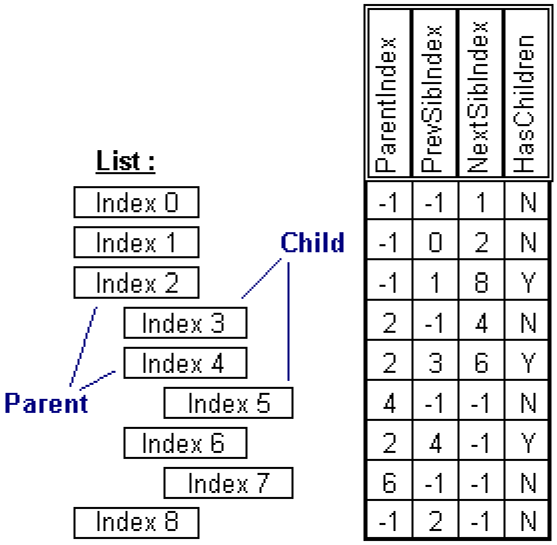
Property Window : No
Des Time : none
Run Time : Read-Write

The CurrPath Property

See also :

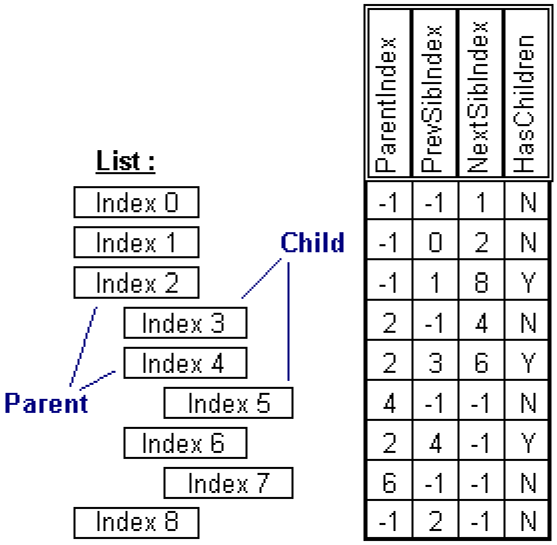
Tree Path Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

String

Purpose :

the AbsPath of the current node

Description :

The CurrPath Property returns a path-like string which uniquely identifies the Current Node. To build this string, the Tree starts with the Current Node and works its way up through the Parent Nodes, adding the Index number for each Node as it goes. Indexes listed in the CurrPath Property are separated by periods '.'. The value of this property can be saved for future use in a string variable. The Tree will allow you to change the Current Node by applying a stored CurrPath string.

Essentially, the CurrPath is (starting at the Root Node) the list of indexes of the Child Nodes which occur within the ancestry of the Current Node.   The CurrPath of the Root Node is always an empty string "".

This property gives the Tree some very powerful functionality.   Since each Node can be uniquely identified by a CurrPath string, the CurrPath of a Node can be stored in a string and then reset later to restore the Tree to that Current Node.

Example :

```
- Setting Values with Code :
Dim I As Integer
Dim S As String
' Save the CurrPath of the Current Node
S = Tree.CurrPath
' Goto the First Node
Tree.MoveCurr = 1
' Count the Number of Siblings
I = 0
While Not Tree.BadMove
    I = I + 1
    Tree.MoveCurr = 4 ' Get Next Node
Wend
' Reset Current Node
Tree.CurrPath = S
MsgBox "Found " + Str$(I) + " Node(s)"
```

Property Access :

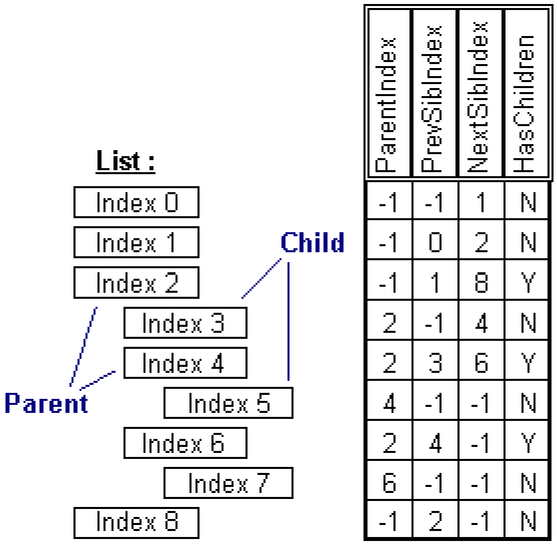
Property Window : No
Des Time : none
Run Time : Read-Only

The CurrString Property

See also :

Item Data Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree

Property Type :

String

Purpose :

main string storage

Description :

The CurrItemText Property is the string data associated with the Current Node. This property is the counterpart to the listbox's ItemText Property Array, and will generally be the Tree's main form of data storage. You may set the CurrItemText Property of a Node at Run-Time and, by virtue of moving to that node's parent, access that same string later through the ItemText Property.

This property provides the primary storage medium for the Tree. While the most useful way to interact with the Tree is through the ItemText, ItemData, and ItemString Properties (pertaining to the Current List of Children), it is also at times useful to know what the Current Node's ItemText, ItemData and ItemString values are.

Example :

```
- Setting Values with Code :
Dim S As String
' Set the String for the Current Node
S = "This is some text."
Tree.CurrString = S
' Move to the Parent Node
Tree.MoveCurr = 5
' Search for S in the Child List and get the Index
Tree.SearchString = S
MsgBox (S + " was found at : " + Str$(Tree.FoundIndex))
```

```
- Getting Values with Code :
Dim S As String
' Add a Child Node to the Tree
Tree.AddItem "This is a Child Node"
' Move to That Node and get the same string
Tree.MoveToChild = Tree.NewIndex
S = Tree.CurrString
```

Property Access :

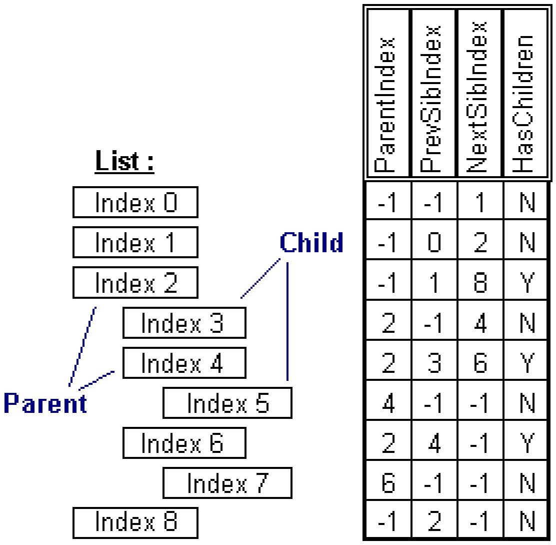
Property Window : No
Des Time : none
Run Time : Read-Write

The DefFlags Property

See also :

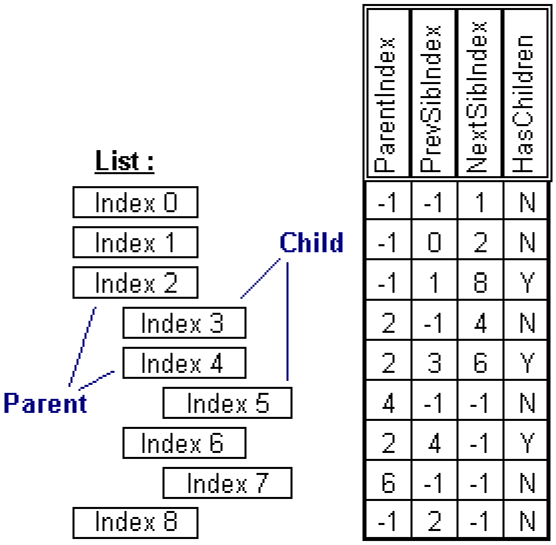
Item Defaults Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

String, Property Array

Purpose :

Default Item Flags

Description :

This property allows you to dynamically alter the display and behaviour of your items in your list easily and efficiently. This propertty is used to automatically use certain flags based upon an item's state (current, selected, parent, etc.). Using flags (either in string or numeric form) you are able to change the following aspects of item display : Border Display, Item Image placement, Text placement, Text 3D, Shadowing, and Highlighting.

Note: The six things you can set defaults for are: Disabled, Selected, Current, Parent Open, Parent Closed, Leaf

Example :

```
- Setting Values with Code :
' Init FileList Defaults
' Inclue File TLC20.BAS
lstFile.BackColor = QBColor(7)
lstFile.TrueShadow = True
lstFile.ControlFlags = "+BRD_SHADOW"
lstFile.ImageBag = lstImages.hWnd
lstFile.DefStateImage(DEFITEM_SELECTED) = "SELECTED"
lstFile.DefFlags(DEFITEM_SELECTED) = "0+HIL_TEXT"
lstFile.DefStateImage(DEFITEM_CURRENT) = "CURRENT"
lstFile.DefFlags(DEFITEM_CURRENT) = "0+HIL_TEXT"
lstFile.DefImage(DEFITEM_PARENTO) = "OPEN"
lstFile.DefImage(DEFITEM_PARENTC) = "CLOSED"
lstFile.DefImage(DEFITEM_LEAF) = "LEAF"
lstFile.DefStyle(DEFITEM_LEAF) = "LEAF"
lstFile.ItemSpacing = 1
```

Property Access :

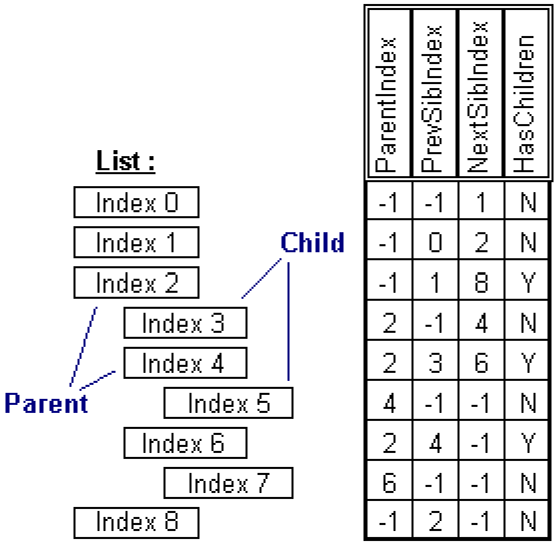
Property Window : No
Des Time : none
Run Time : Read-Write

The DefHeight Property

See also :

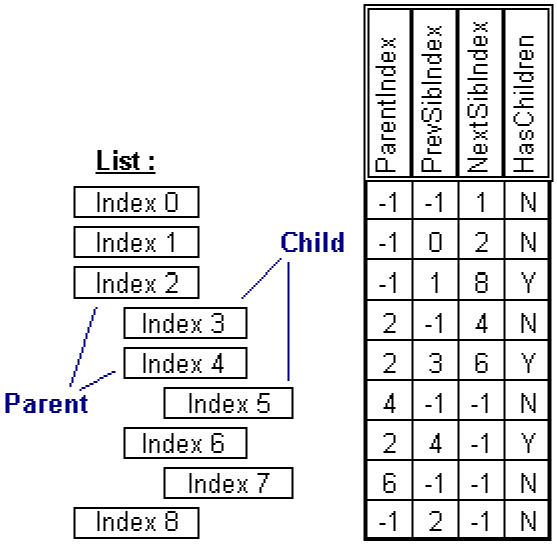
Item Defaults Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Integer, Property Array

Purpose :

Default Item Height

Description :

Specifies the height of an individual item.   If this property is not set, the control will automatically calculate the item's size based on the text, style, and image.

Example :

```
- Setting Values with Code :
lst.DefHeight = 36

- Getting Values with Code :
Dim I as integer
I = lst.DefHeight
```

Property Access :

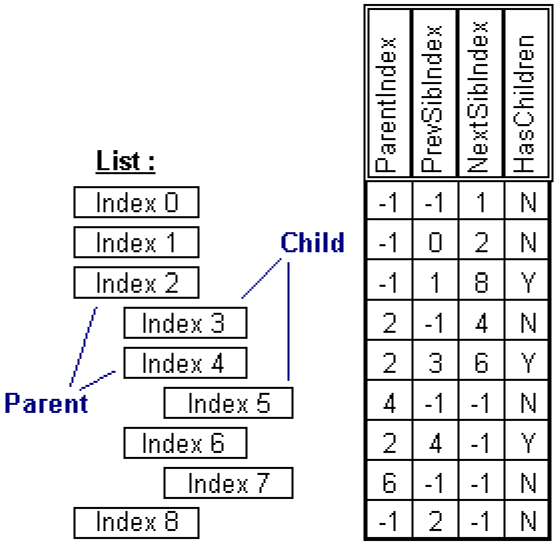
Property Window : No
Des Time : none
Run Time : Read-Write

The DefImage Property

See also :

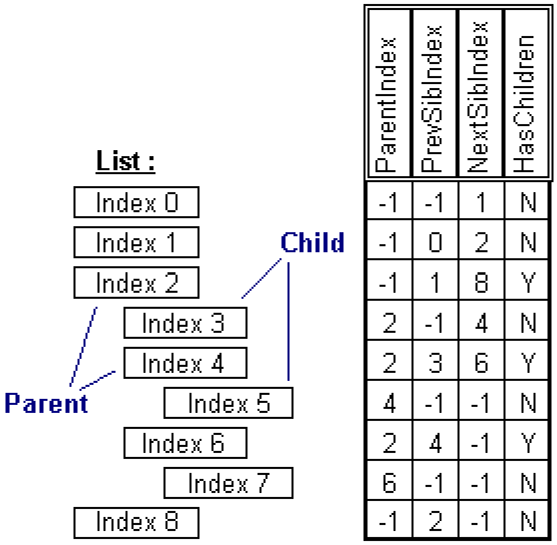
Item Defaults Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

String, Property Array

Purpose :

Default Item Image

Description :

This property allows you to specify an image that will be displayed with an item. This propertty is used to automatically display certain images based upon an item's state (current, selected, parent, etc.). If an item's width and height properties are not set (i.e. they are -1), the item will be automatically sized to include the displayed image.

Note that when using this property, the control which is to display the images must be connected to an ImageBag via the ImageBag property. The value of this property should be the name or index of the image to be used as it occurs in the connected ImageBag.

Example :

```
See example in DefFlags property
```

Property Access :

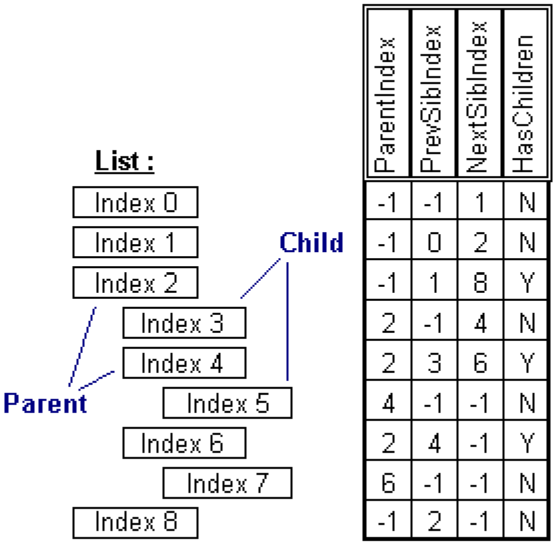
Property Window : No
Des Time : none
Run Time : Read-Write

The DefStateImage Property

See also :

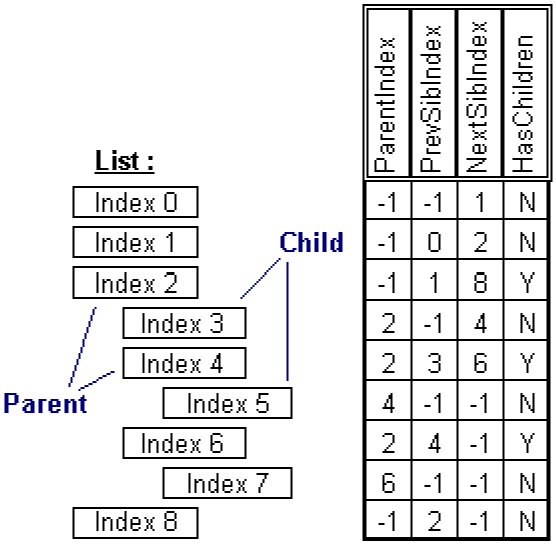
Item Defaults Properties

Applies To :



List :	ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
Index 0	-1	-1	1	N
Index 1	-1	0	2	N
Index 2	-1	1	8	Y
Index 3	2	-1	4	N
Index 4	2	3	6	Y
Index 5	4	-1	-1	N
Index 6	2	4	-1	Y
Index 7	6	-1	-1	N
Index 8	-1	2	-1	N

TabRas



List :	ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
Index 0	-1	-1	1	N
Index 1	-1	0	2	N
Index 2	-1	1	8	Y
Index 3	2	-1	4	N
Index 4	2	3	6	Y
Index 5	4	-1	-1	N
Index 6	2	4	-1	Y
Index 7	6	-1	-1	N
Index 8	-1	2	-1	N

TSTree

Property Type :

String, Property Array

Purpose :

Default State Image

Description :

This property allows you to specify an image that will be displayed outside and to the left of an item.   This propertty is used to automatically display certain images based upon an item's state (current, selected, parent, etc.).

Note that when using this property, the control which is to display the images must be connected to an ImageBag via the ImageBag property.   The value of this property should be the name or index of the image to be used as it occurs in the connected ImageBag.

Example :

`See example in DefFlags property`

Property Access :

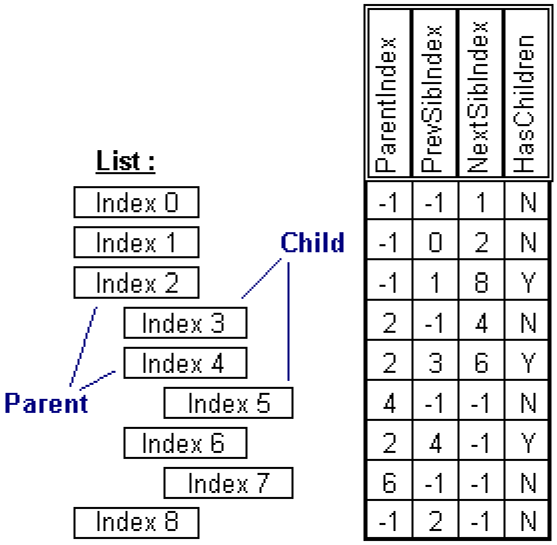
Property Window : No
Des Time : none
Run Time : Read-Write

The DefStyle Property

See also :

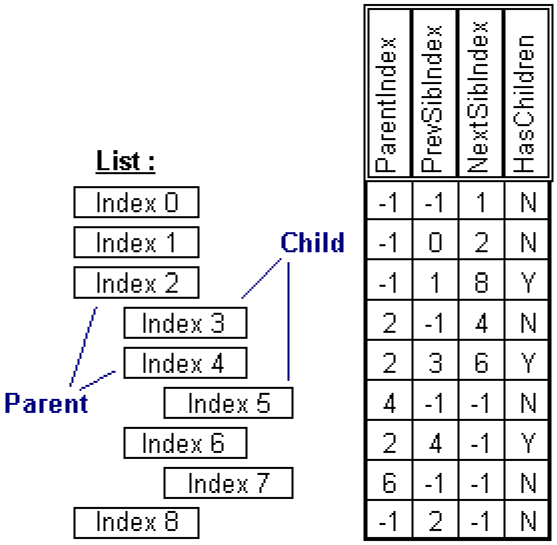
Item Defaults Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

String, Property Array

Purpose :

Default Item Style

Description :

This property allows you to specify a style with which to display an item.   This propertty is used to automatically use certain styles based upon an item's state (current, selected, parent, etc.).   An item's style may dictate that item's size, font, color, border, and also contributes to the item's ItemFlags property.

Note that when using this property, the control which is to display the images must be connected to a StyleBag via the StyleBag property.   The value of this property should be the name or index of the style to be used as it occurs in the connected StyleBag.

Example :

`See example in DefFlags property`

Property Access :

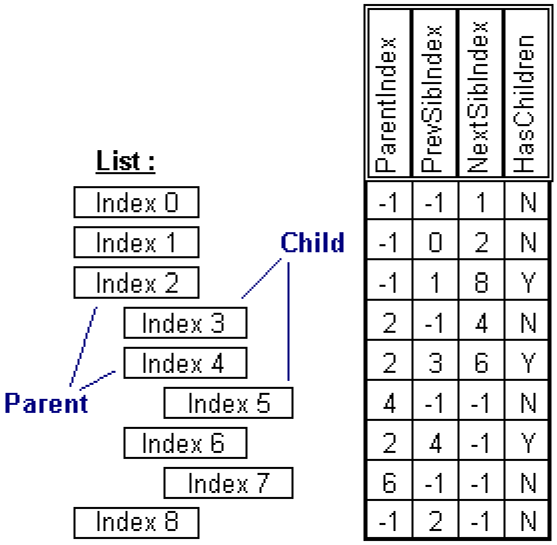
Property Window : No
Des Time : none
Run Time : Read-Write

The DefWidth Property

See also :

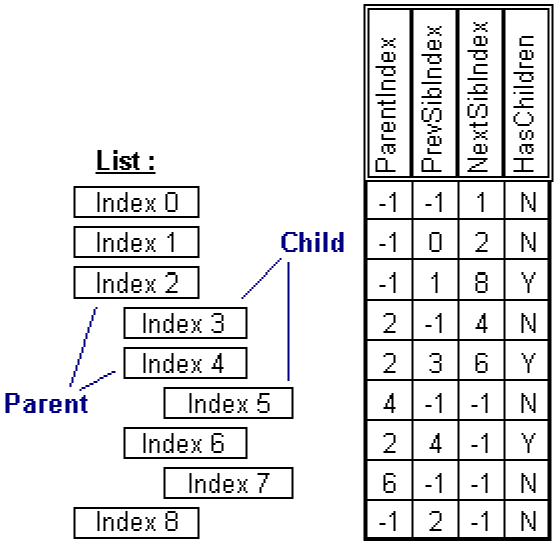
Item Defaults Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Integer, Property Array

Purpose :

Default Item Width

Description :

Specifies the width of an individual item.   If this property is not set, the control will automatically calculate the item's size based on the text, style, and image.

Example :

```
- Setting Values with Code :
lst.DefWidth = 360
```

```
- Getting Values with Code :
Dim I as integer
I = lst.DefWidth
```

Property Access :

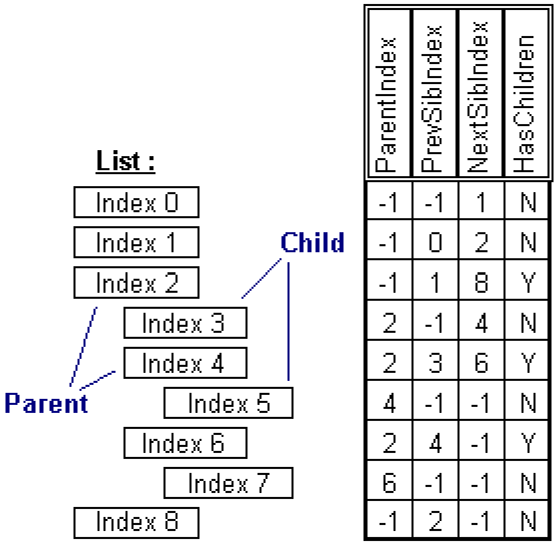
Property Window : No
Des Time : none
Run Time : Read-Write

The DisplayType Property

See also :

List Display Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree

Property Type :

Boolean

Purpose :

controls the way embedded formatting codes are handled when displaying items

Description :

The DisplayType Property controls the way embedded formatting codes are handled when displaying items. This property accepts a value between 0 and 2. When set to **0 (Regular)**, the list will display the text in each item as is. If there are any formatting codes within the text, they will be displayed as well. If set to **1 (Formatted)**, the list interprets all formatting codes and displays the text with the formats indicated by the codes (font attributes, color, etc.). The last value, **2 (RawText)**, strips the text of all formatting codes before displaying it. The text will not be formatted and the formatting codes will not be displayed with this setting.

This property can be useful when you know that you will not have any embedded formatting commands in your display text. If this is the case, you can set the the DisplayType Property to 0 (Regular DisplayType) and benefit from a much quicker list display.'Formatted" and 'RawText" DisplayTypes are available as well. When using formatting commands in your text, you will generally want to use one of these two DisplayTypes. The Formatted Display makes full use of all formatting commands and can provide a very meaningful display to the user; However, it will take a little extra time to process each formatting command. The RawText Display is provided so that you can switch between displaying the fully formatted text and just the text itself. This feature can provide a very visual effect when setting the DisplayType to Formatted when the list has the focus, and back to RawText when it loses focus.

Example :

```
- Setting Values with Code :
' Set the DisplayType to show just the unformatted text
List.DisplayType = 2

- Getting Values with Code :
Dim I As Integer
I = List.DisplayType
```

Property Access :

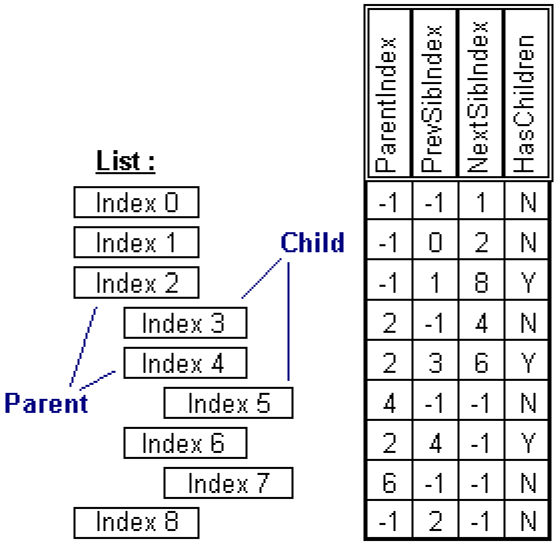
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The DrawMethod Property

See also :

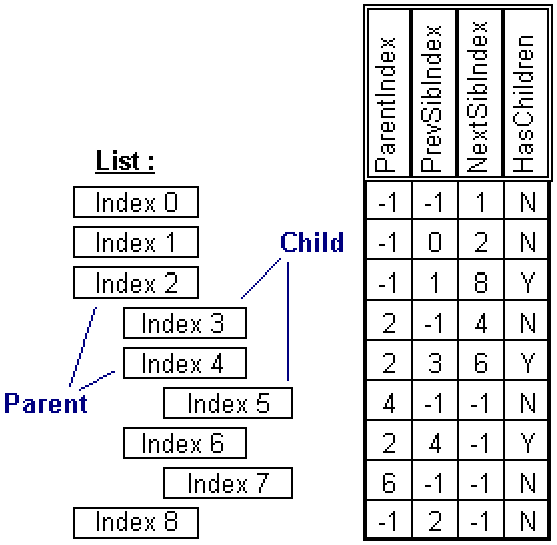
List Display Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList

Property Type :

Integer (Enum)

Purpose :

gives you control over how the List will draw itself to the screen when an update is required.

Description :

When the DrawMethod property is set to **1 (Smooth Drawing)**, the screen update process provides the List with a more continuous and smooth scrolling effect when being scrolled with the scrollbar or keyboard. If set to **0 (Normal Drawing)**, unnecessary redrawing may be noticed.

Example :

```
- Setting Values with Code :
```

```
List.DrawMethod = 0 ' Normal
List.DrawMethod = 1 ' Smooth

- Getting Values with Code :
Dim I As Integer
I = List.DrawMethod
```

Property Access :

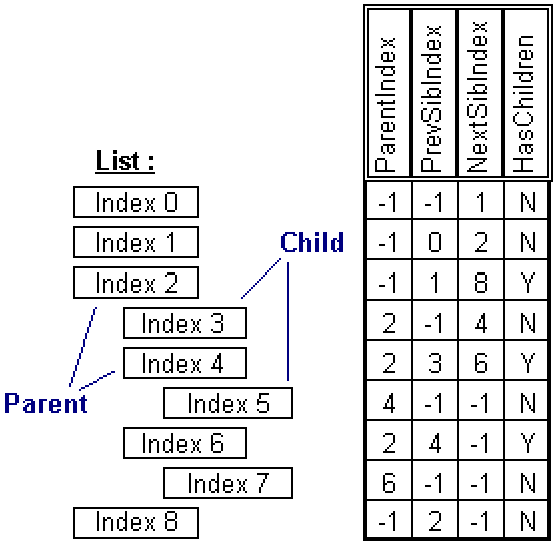
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The EscChar Property

See also :

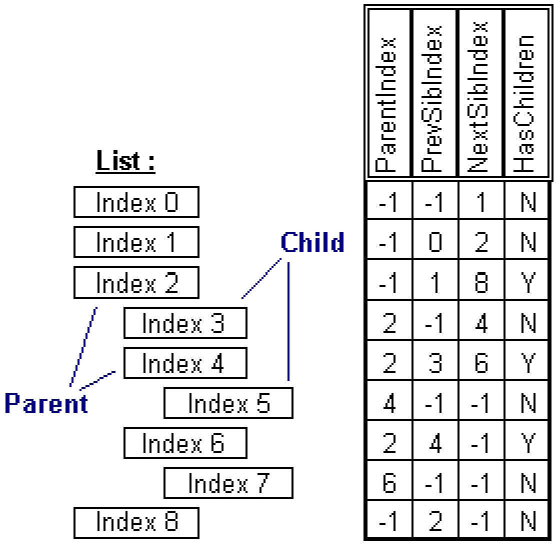
Control Properties

Applies To :



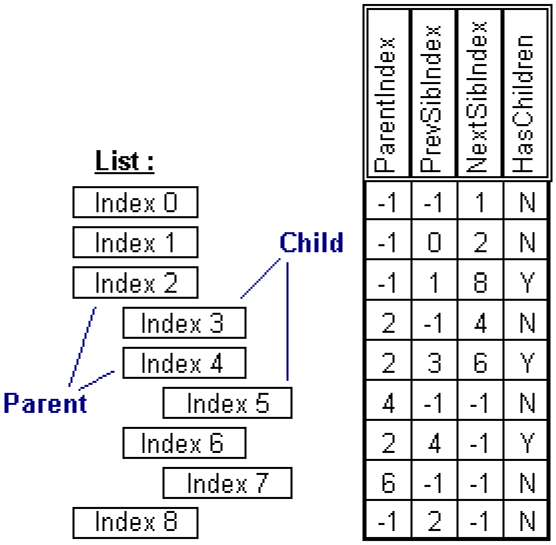
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



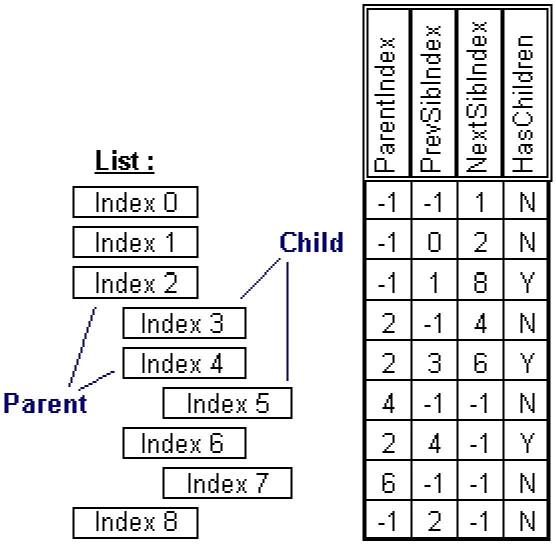
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

String, single character

Purpose :

special character used to embed formatting codes in text

Description :

The EscChar Property defines the single character used to delimit the embedded formatting codes.   The default for this property is the vertical bar "|".   In most cases, the formatting codes require the EscChar to appear before and after the embedded formatting code.
This property allows you to customize the character used to delimit embedded formatting codes.   This can be useful when you suspect that you will be displaying the vertical bar as text in your list.   In such cases, you may change the EscChar to be some unused character.   Good candidates would be characters whose ASCII value are between 1 and 7.   You can assign these values to the EscChar Property with the VB Chr$ function at Run-Time.   Note : Do Not Ever set the EscChar Property to character zero, i.e. Chr$(0).

Example :

```
- Setting Values with Code :
Dim S As String
' Put char 1 in the string
S = Chr$(1)
' Set the EscChar to char 1
List.EscChar = Chr$(1)
' Add a Bolded Item to the List
List.AddItem (S + "B" + "This is Bold Text.")
```

```
- Getting Values with Code :
Dim S As String
S = List.EscChar
```

Property Access :

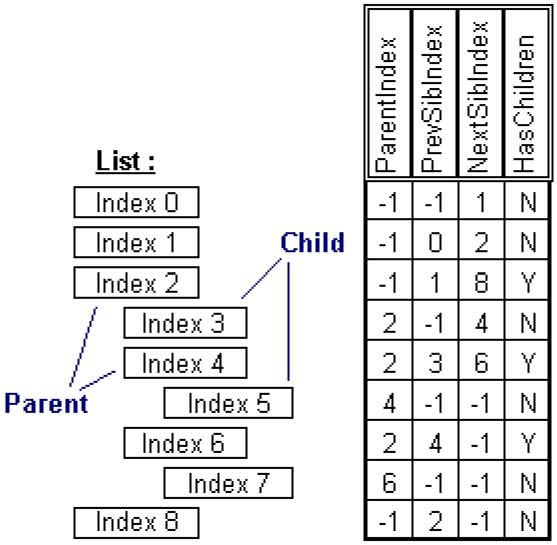
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The ExtraString Property

See also :

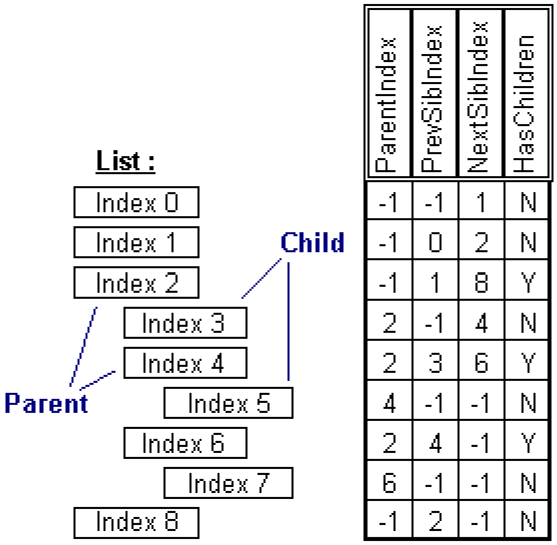
Item Data Properties

Applies To :



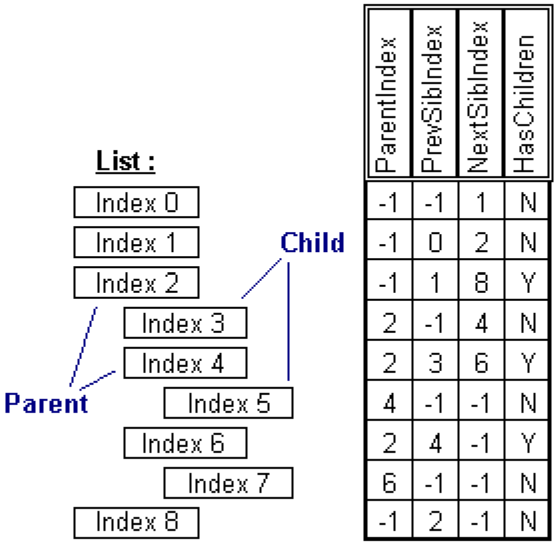
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



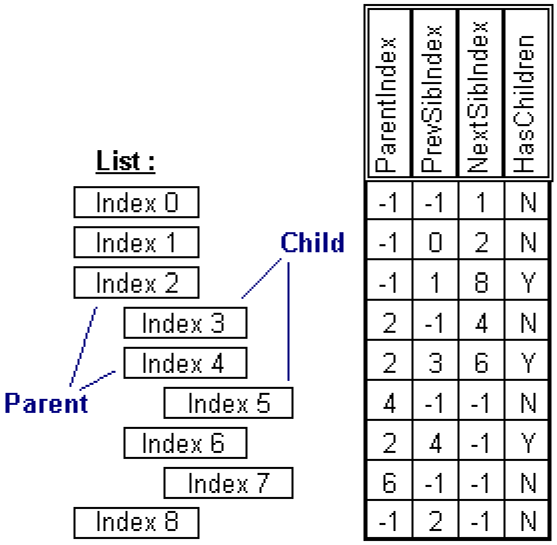
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList



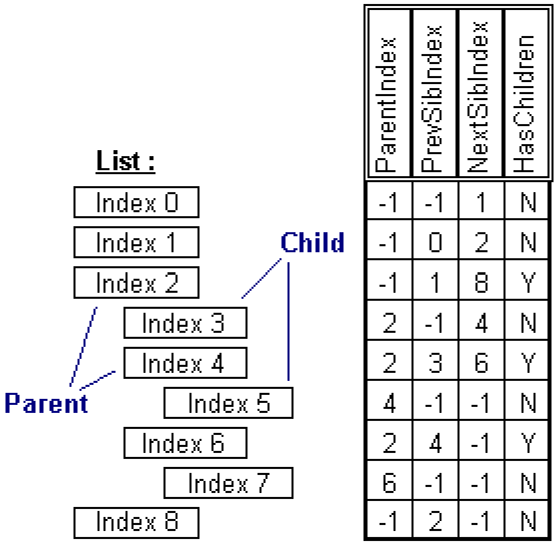
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

String, Property Array

Purpose :

(see ItemString Property)

Description :

This property, similar to the ItemData property, can be used to associate each item in a List with extra data stored elsewhere (for example, the ExtraString may hold a BookMark string for each record in a table). The ExtraString may also be used to store your data, with the ItemText property displaying only what you want the user to see.

This property is a Property Array. That is, when setting and getting values with this property, you must also supply an Index (inside the parentheses) in order to tell the List which Item you are talking about. The property's Index is Zero-Based, meaning that the first Index is 0, the second is 1, etc.,.

Example :

```
- Setting Values with Code :
Dim S As String
S = "This Text is not going to be Displayed"
List.ExtraString( 0 ) = S

- Getting Values with Code :
Dim S As String
S = List.ExtraString( 0 )
```

Property Access :

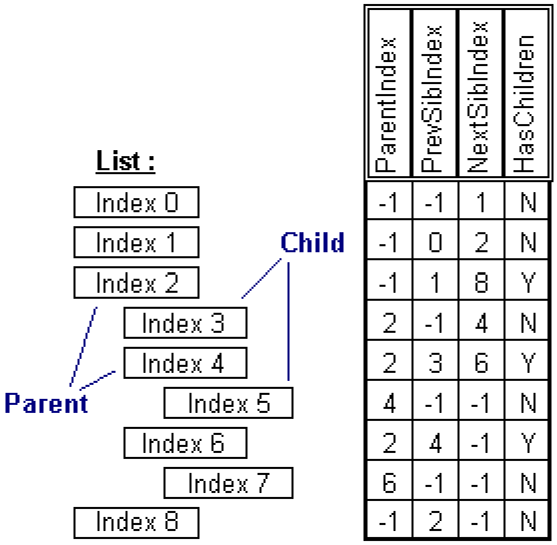
Property Window : No
Des Time : none
Run Time : Read-Write

The FileAction Property

See also :

File/Dir Viewing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList

Property Type :

Integer (Enum)

Purpose :

load, save, or delete filename

Description :

The FileAction Property is used in conjunction with the FileName Property. When setting the FileAction property, the FileName Property indicates the name of the file to be read, written or deleted from the disk. As soon as the FileAction Property is set, the list performs one of the following actions :
1 - Read FileName and display contents of file in the list
2 - Write the contents of the list to the file specified by FileName (if FileName does not exist, the list will create it)
3 - Delete the file specified by FileName

This property relieves much dreariness from your programs by handling TextFile I/O for you. Any text file can be loaded and inspected line-by-line in a program. The program may then modify the file as it is in the listbox, and then rewrite the file to the disk or even delete a temporary file.

Example :

```
- Setting Values with Code :
' Load and Display the AUTOEXEC.BAT File
List.FileName = "C:\AUTOEXEC.BAT"
If (List.FileExists = True) Then
    List.BatchMode = True
    List.FileAction = 1 ' Read the File
    List.BatchMode = False
End If

- Getting Values with Code :
Dim I As Integer
I = List.FileAction ' Always Returns Zero!
```

Property Access :

Property Window : No
Des Time : none

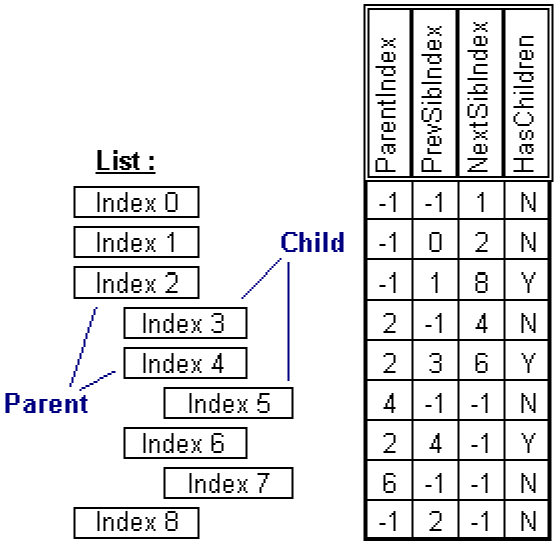
Run Time : Read-Write

The FileExists Property

See also :

File/Dir Viewing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList

Property Type :

Boolean

Purpose :

determine if filename exists

Description :

The FileExists Property is used in conjunction with the FileName Property. The value of the FileExists Property is True if FileName actually exists on the disk, or False otherwise. The FileName Property accepts a valid file name and, optionally, a path. If no path is specified in FileName, the list will attempt to locate the file in the following directories :

1 - the Current Directory of the Current Drive
2 - the Windows Directory
3 - the Windows\System Directory
4 - the "App.Path" of where your program was loaded from
5 - the DOS Path
6 - all of the directories mapped on a network

This property allows you to see if a given filename actually exists somewhere on the disk before trying to read it using the FileAction Property.

Example :

```
see example in FileAction Property
```

Property Access :

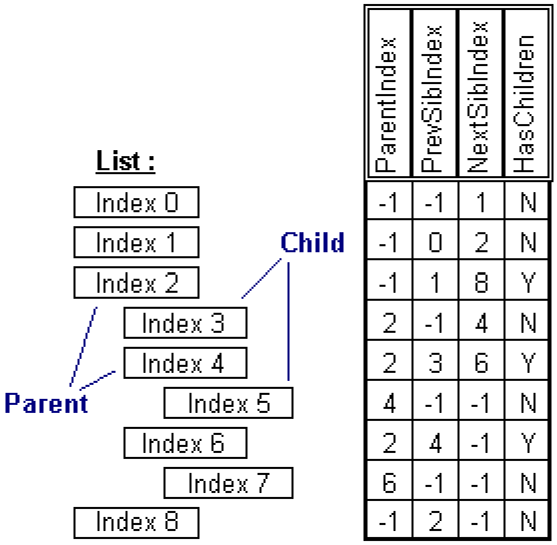
Property Window : No
Des Time : none
Run Time : Read-Write

The FileName Property

See also :

File/Dir Viewing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList

Property Type :

String

Purpose :

the filename of the file to load or save

Description :

The FileName Property is used in conjunction with the FileAction and FileExists Properties. For the FileAction property, the FileName Property indicates the name of the file to be read, written or deleted from the disk. The value of the FileExists Property is True if FileName actually exists on the disk, or False otherwise. The FileName Property accepts a valid file name and, optionally, a path. If no path is specified in FileName, the list will attempt to locate the file in the following directories :

1 - the Current Directory of the Current Drive
2 - the Windows Directory
3 - the Windows\System Directory
4 - the "App.Path" of where your program was loaded from
5 - the DOS Path
6 - all of the directories mapped on a network

This property provides for the functionality found in the FileAction and FileExists Properties.

Example :

```
see example in FileAction Property
```

Property Access :

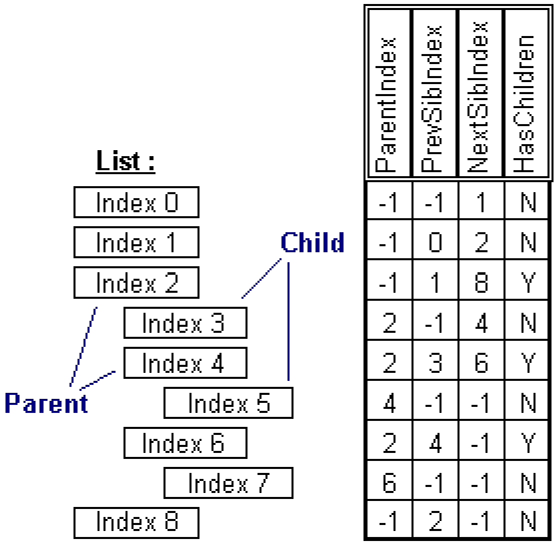
Property Window : No
Des Time : none
Run Time : Read-Write

The FoundIndex Property

See also :

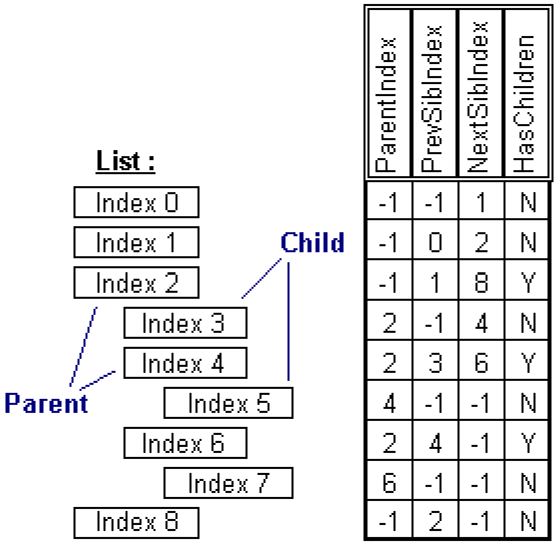
List Searching Properties

Applies To :



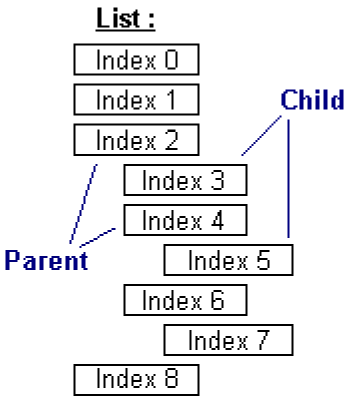
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



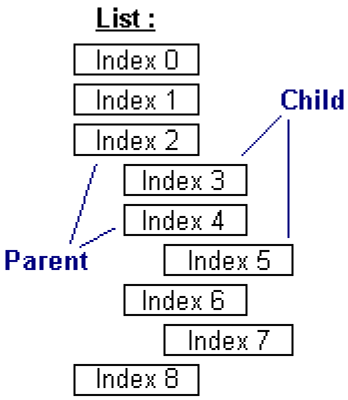
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList



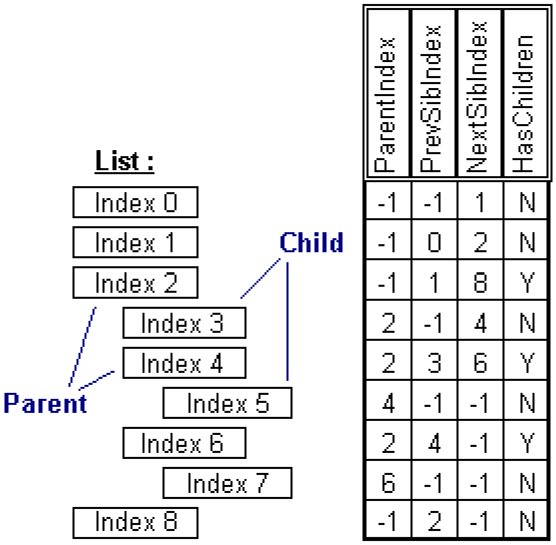
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



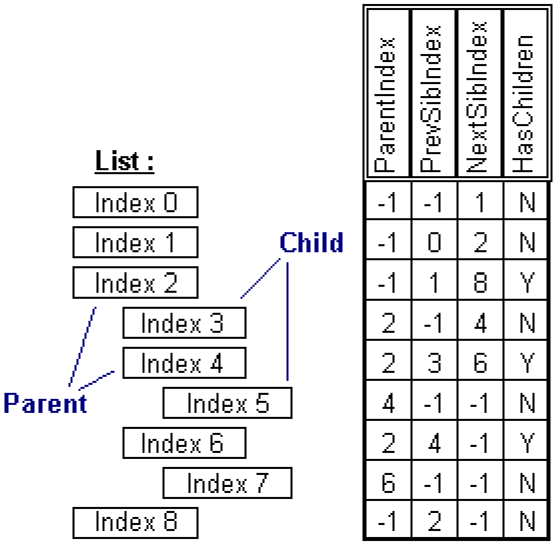
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse



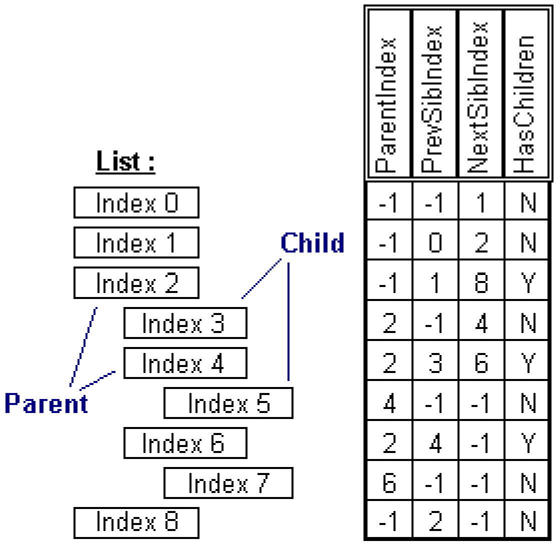
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



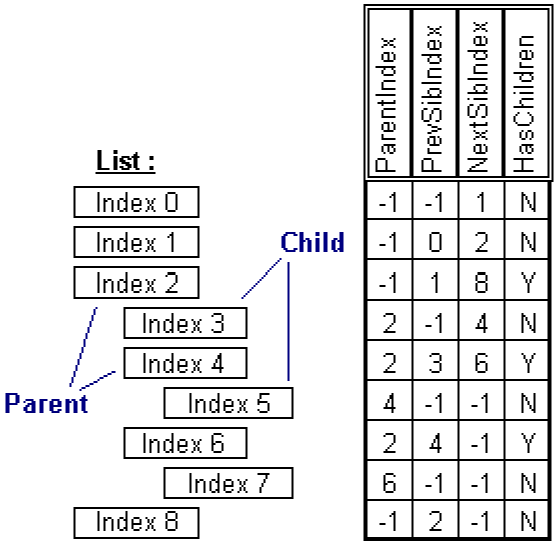
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Long

Purpose :

This property is used to check the results of searching the List with the StartSearch and SearchString properties.

Description :

The FoundIndex Property is used with the StartSearch and SearchString Properties. When using the SearchString Property, the List will begin searching for the specified string at Index specified by the StartSearch property. If the string is found in the List, FoundIndex will contain the index at which the string was found. If the string was not found in the List, then FoundIndex will be set to -1.

Example :

```
- Getting Values with Code :
' Find All Instances of a String in the List
Dim S As String
S = 'USA"
List.StartSearch = 0
List.SearchString = S
While List.FoundIndex <> -1
    List.StartSearch = List.FoundIndex + 1
    List.SearchString = S
Wend
```

Property Access :

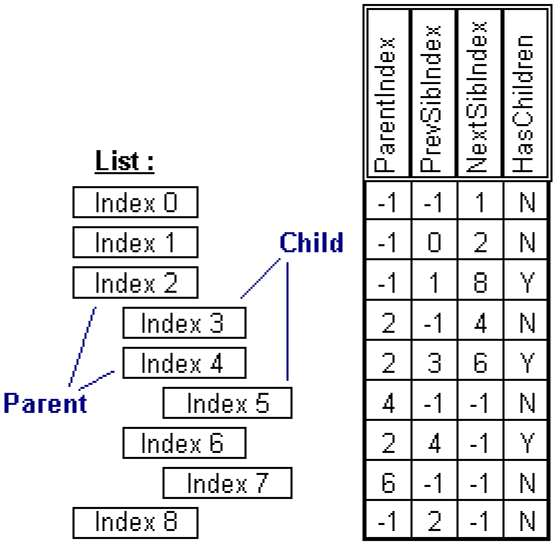
Property Window : No
Des Time : none
Run Time : Read-Only

The HasChildren Property

See also :

Tree Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree

Property Type :

Boolean, Property Array

Purpose :

indicates whether a certain item has child items

Description :

The HasChildren Property is a boolean property that returns True if an Item has subitems indented underneath of it, or False otherwise.

When maintaining a tree structure with a List, you may not always want to have all items and subitems displayed at once. The HasChildren Property allows you to determine if an Item already has subitems underneath of it, or that its subitems need to be determined and added to the List.

Example :

```
- Getting Values with Code :
' Check if we Need to Load FilesNames for this Directory
If Not List.HasChildren( List.ListIndex ) Then
    Call LoadFileNames( List.ListIndex )
End If
```

Property Access :

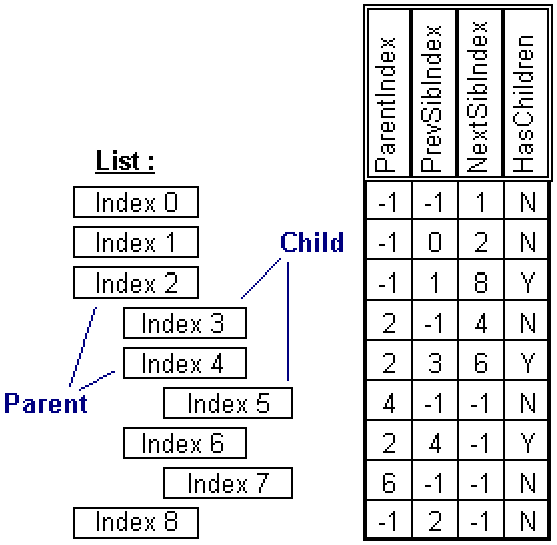
Property Window : No
Des Time : none
Run Time : Read-Write

The HighlightStyle Property

See also :

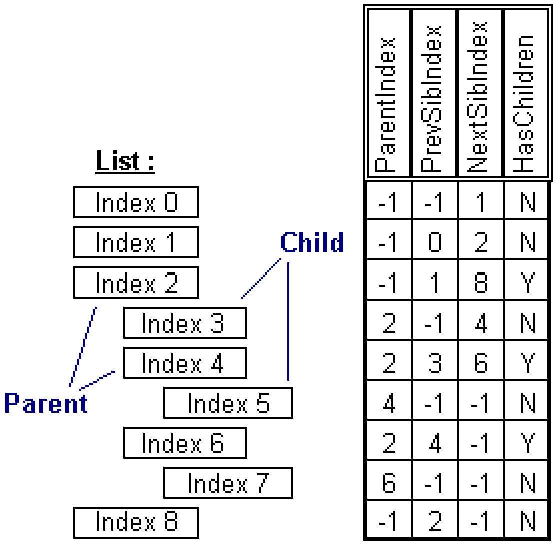
List Display Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList

Property Type :

Boolean

Purpose :

determine the type of highlighting to use for an item

Description :

The HighlightStyle Property controls how the list will display currently selected items. Normally, a listbox will highlight the current item by drawing a filled-in highlight rectangle (usually in a blue color) to fill the whole item in the display. The user can change the current item by clicking an item in the list, and the program can change the current item by setting the ListIndex of the list. By setting the HighlightStyle Property, the list will display it's current item by highlighting in one of three styles :

0 - None : This style performs no highlighting. The only indication of the current item is the focus rectangle drawn when the list has the focus.

1 - Highlight Text : This style forces the current item's text to be drawn with a blue background (the highlight color is actually determined by system's highlight color setting). This has the effect of highlighting only the text displayed in an item.
2 - Highlight Item : This style is the same as the standard highlighting behavior for listboxes. That is, highlighting the whole item in the display.

The reason for including this property is simply to have some variety in the display styles available. Furthermore, when using formatted text in a CTree or using your own visual cue as to the current item (such as Bolding the text of the current item), it is nice to have the HighlightStyle Property set to 0 (None) to allow all the coloring effects to be visible.

Example :

```
- Setting Values with Code :
' Set Style to Highlight Text Only
List.HighlightStyle = 1

- Getting Values with Code :
Dim I As Integer
I = List.HighlightStyle
```

Property Access :

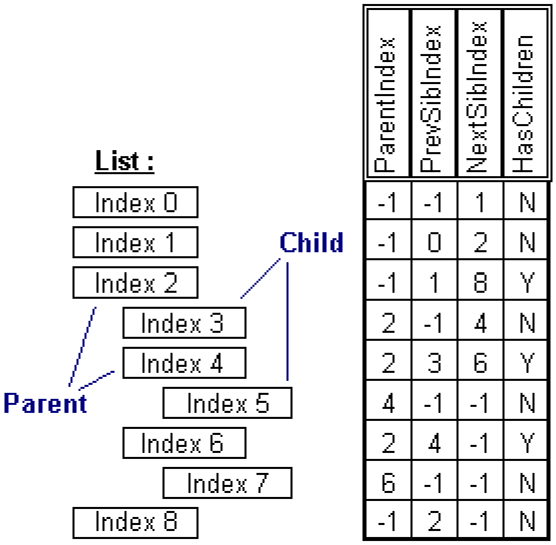
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The hWnd Property

See also :

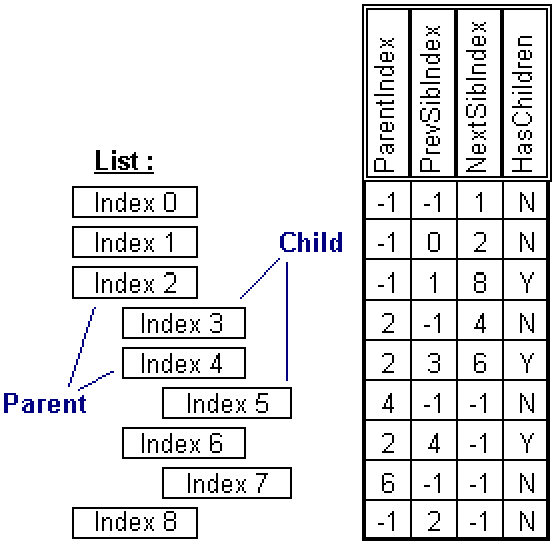
Control Properties

Applies To :



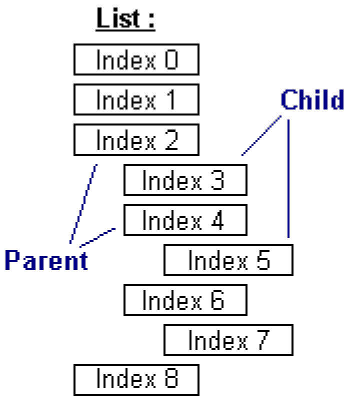
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



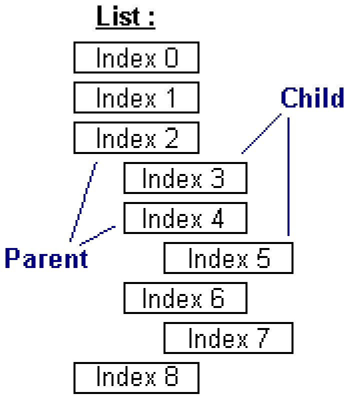
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList



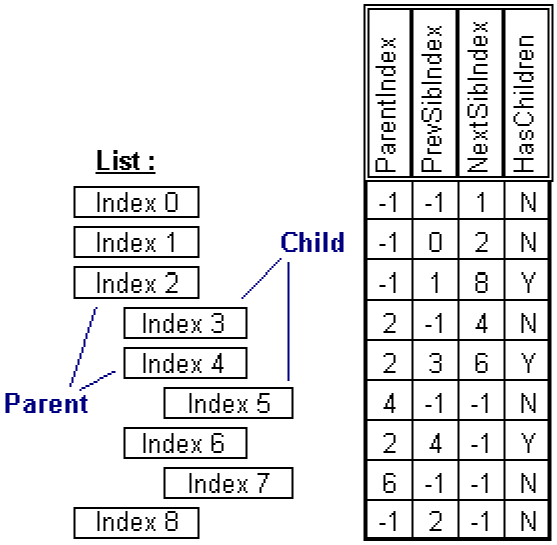
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



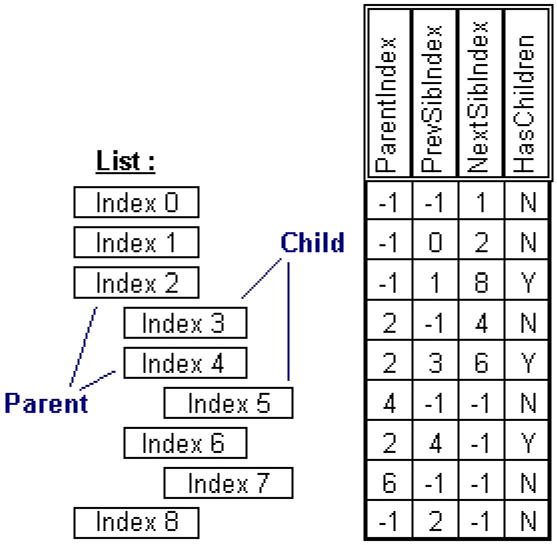
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse



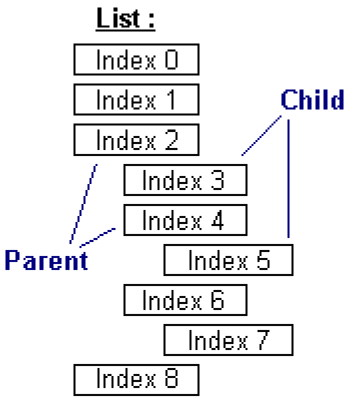
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



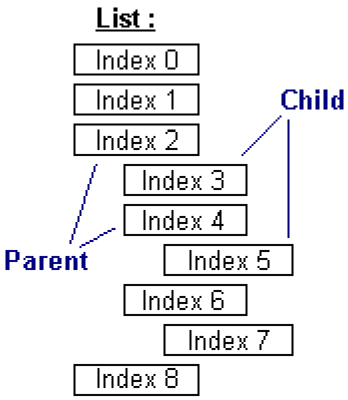
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



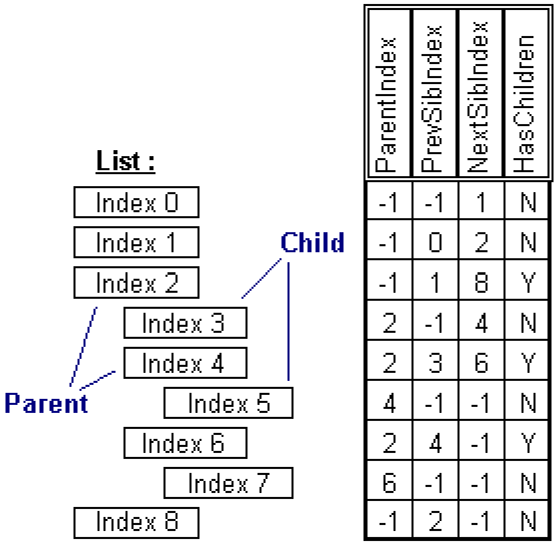
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Integer

Purpose :

Window Handle of Control

Description :

The hWnd Property contains the "Window Handle" of the ListBox Window. This property can be used to uniquely identify the partcular instance of that control. This property is also used in combination with the ImageBag and StyleBag properties. This acts to enable the ImageBag and StyleBag controls as "servers" for other controls.

Note: The Templar List Controls do not subclass the standard Window's ListBox and, therefore, will not respond to the standard ListBox messages sent with the SendMessage API.

Example :

```
- Getting Values with Code :
Dim I As Integer
' Get ImageBag's hWnd
I = ImageBag1.hWnd
' Connect to Label
TSLabel1.ImageBag = I
```

Property Access :

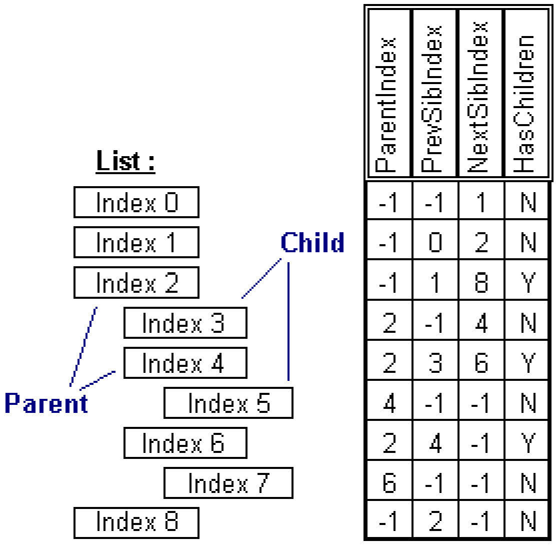
Property Window : No
Des Time : Read-Only
Run Time : Read-Only

The Image Property

See also :

Image Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag

Property Type :

Picture, Property Array

Purpose :

Specifies a handle to a bitmap; the handle is provided by the system.

Description :

The array of Image Properties along with the ImageHeight, ImageWidth, ImageName, and ImageZoom is what makes up the ImageBag control.

Example :

```
- Setting Values with Code :
Imagebag1.ImageName(1) ="Folder Closed"
Imagebag1.ImageWidth(1) = 25
Imagebag1.ImageHeight(1) = 25
Imagebag1.ImageZoom(1) = 1
Imagebag1.Image(1) = LoadPicture(cmdialog1.filename)
' Use the ImageName instead of its rray index
ctrl.ImageBag = ImageBag1.Hwnd
ctrl.ItemImage(0) = "Folder Closed"
```

Property Access :

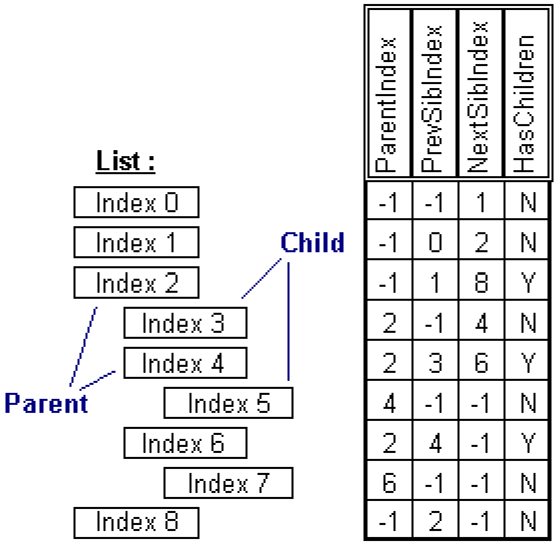
Property Window : No
Des Time : none
Run Time : Read-Write

The Image1..9 Property

See also :

Image Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree

Property Type :

Picture

Purpose :

images to be displayed with items

Description :

The Image properties are placeholders to use for displaying images in the list items using the ItemImage Property. There are nine of these Image properties named Image1, Image2, Image3, etc.,. These properties are generally set at Design-Time and referenced by the ItemImage from within your code at Run-Time. These Image properties are currently limited to containing only bitmaps. The dimensions of Images displayed with Items in the List are determined by the ImageWidth and ItemHeight Properties.

These properties offer you a convenient way to manage the bitmaps used in your list. Images can be easily loaded in your List with the Properties Window at Design-Time. At Run-Time, an Image will be displayed when an Item in the List uses that Image by having an image number in the ItemImage Property.

Example :

```
- Setting Values with Code :
' Load a bitmap from Disk into Image5
List.Image5 = LoadPicture( 'C:\APPIMAGES\Pic1.BMP' )

- Getting Values with Code :
' Get a bitmap from Image5
Image1.Image = List.Image5
```

Property Access :

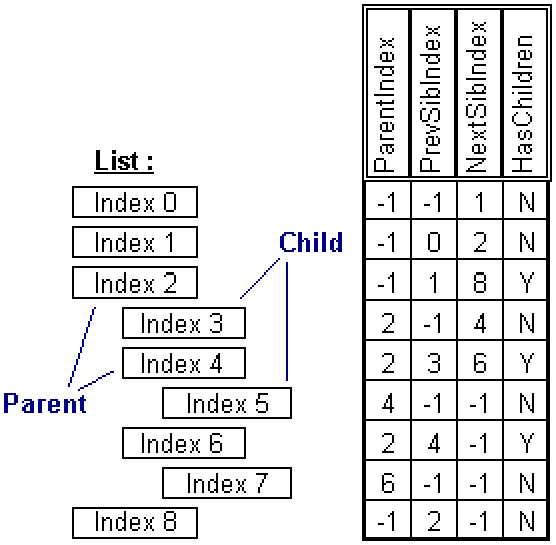
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The ImageBag Property

See also :

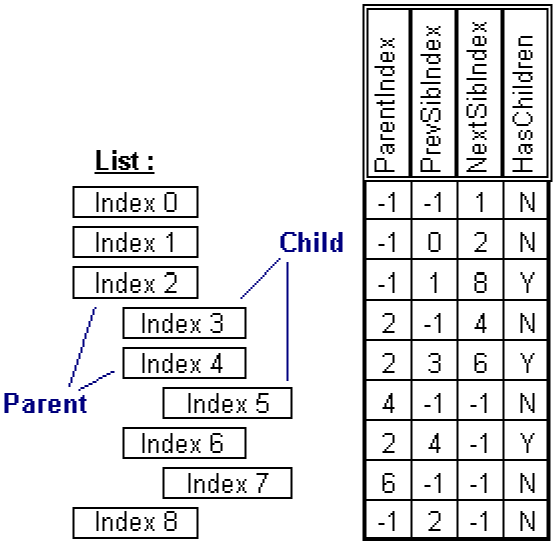
Control Properties

Applies To :



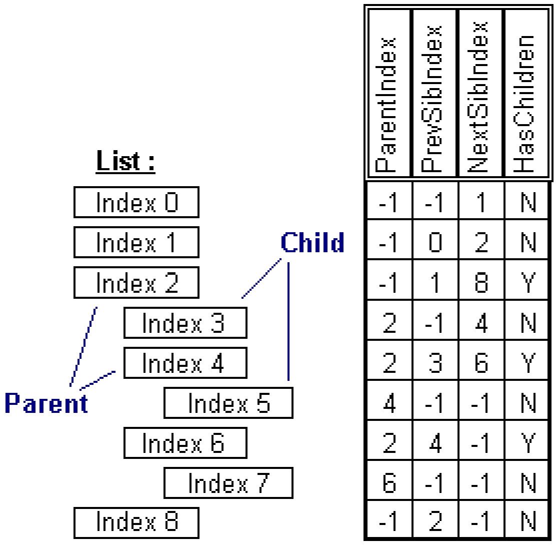
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :
Integer

Purpose :
hWnd of a Source ImageBag

Description :
This property identifies the ImageBag which will serve as the source for all the images displayed within the control. This is much like connecting to a Data Control, except that you are identifying the ImageBag by its window handle (hWnd) rather than its name.

During design-time, you are able to set the ImageBag by entering its name in the properties window. The control will immediately identify the ImageBag's presence and store its window handle. When the control is unloaded and saved to disk at design-time, it stores the ImageBag's name and reconverts it the next time it is loaded.

Example :
```
- Setting Values :
' Set the ImageBag
ctrl.ImageBag = ImageBag1.hWnd

- Getting Values :
Dim I As Integer
' Get the hWnd of the Connected ImageBag
I = ctrl.ImageBag
```

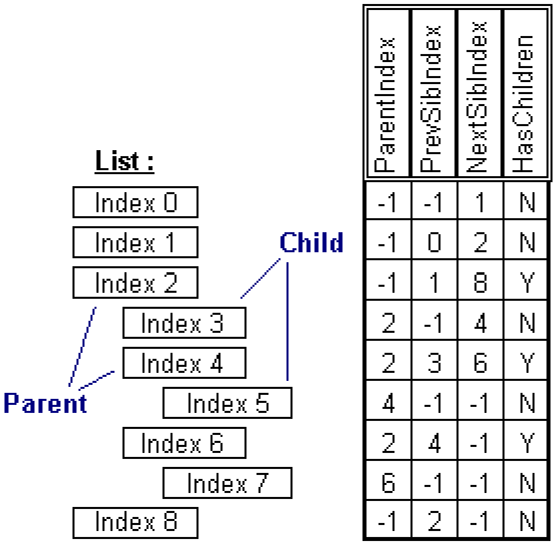
Property Access :
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The ImageHeight Property

See also :

Image Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag

Property Type :

Integer, Property Array

Purpose :

the height to use when displaying an images for items

Description :

The ImageHeight Property specifies the height (in pixels) to use for displaying images via the ItemImage Property. The Image will be Indented with the text if the ItemIndent Property is used. The Image will be displayed in a rectangle which is ItemHeight( ) pixels high and ImageWidth( ) pixels wide.

Example :

```
- Setting Values :
ImageBag1.ImageHeight(0) = 32         ' As tall as an icon
```

Property Access :

Property Window : No
Des Time : none
Run Time : Read-Write

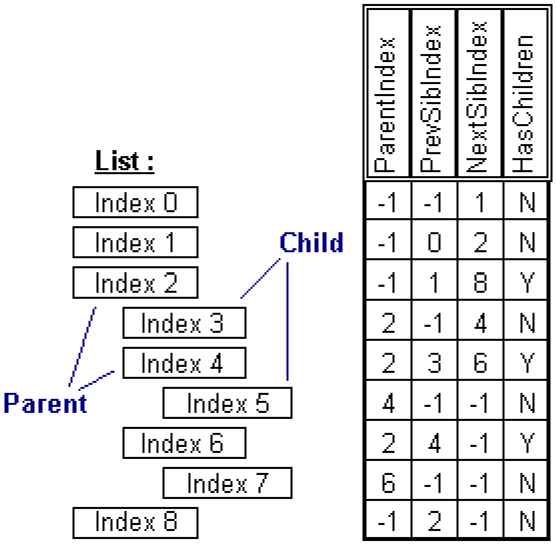
The ImageName Property

See also :

Image Properties

StyleName

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag

Property Type :

String, Property Array

Purpose :

Make code more readable by referrencing images by name from the ImageBag.

Description :

Each item in the ImageBag can hold an Image, ImageHeight, ImageWidth, ImageZoom, and an ImageName. ImageName can help you keep track of images in a better way. Perhaps you have an image of a folder that is closed and a folder that is open. Instead of remembering that the folder closed image is at index 0, you can just assign the ImageName( ) to be Folder Closed. In code, you can then referrence it by name as in the example.

Example :

```
see example in Image( ) Property
```

Property Access :

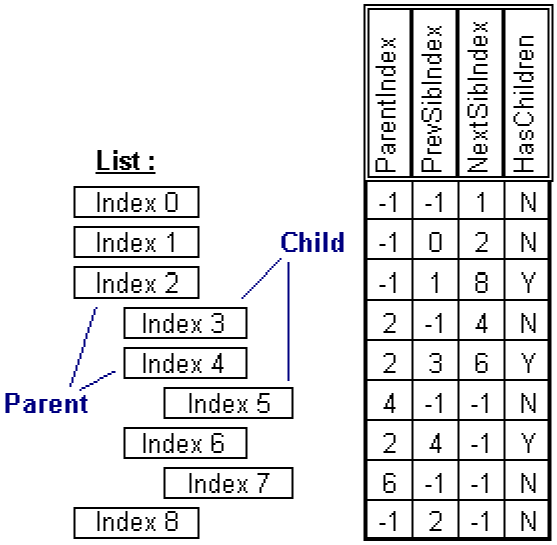
Property Window : No
Des Time : none
Run Time : Read-Write

The ImageWidth Property

See also :

Image Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree

Property Type :

Integer

Purpose :

the width to use when displaying an images for items

Description :

The ImageWidth Property specifies the width (in pixels) to use for displaying images via the ItemImage Property. If the ItemImage Property for an Item is between 1 and 9 (representing a valid Image Number), the list will display the item with an image to the left-hand side of any text. The Image will be Indented with the text if the ItemIndent Property is used, and the Image will be displayed in a rectangle which is ItemHeight pixels high and ImageWidth pixels wide.

This property serves to help maintain consistent dimensions for all images displayed in the list. When creating bitmaps for your list, try to make them all the same width and height. This will provide the cleanest looking display when using the Image properties.

Example :

```
- Setting Values with Code :
' Set Width and Height to Display 20x20 Bitmaps
List.ItemHeight = 20
List.ImageWidth = 20

- Getting Values with Code :
Dim I As Integer
I = List.ImageWidth
```

Property Access :

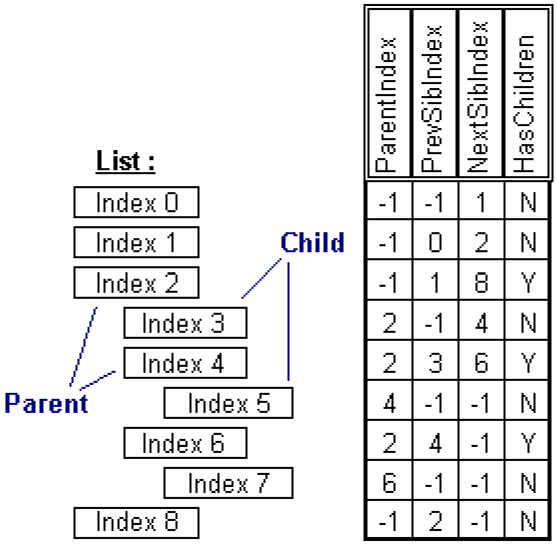
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The ImageWidth Property

See also :

Image Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag

Property Type :

Integer, Property Array

Purpose :

the width to use when displaying an images for items

Description :

The ImageWidth Property specifies the Width (in pixels) to use for displaying images via the ItemImage Property. The Image will be Indented with the text if the ItemIndent Property is used. The Image will be displayed in a rectangle which is ItemHeight( ) pixels high and ImageWidth( ) pixels wide.

Example :

```
- Setting Values :
ImageBag1.ImageWidth(0) = 32          ' As wide as an icon
```

Property Access :

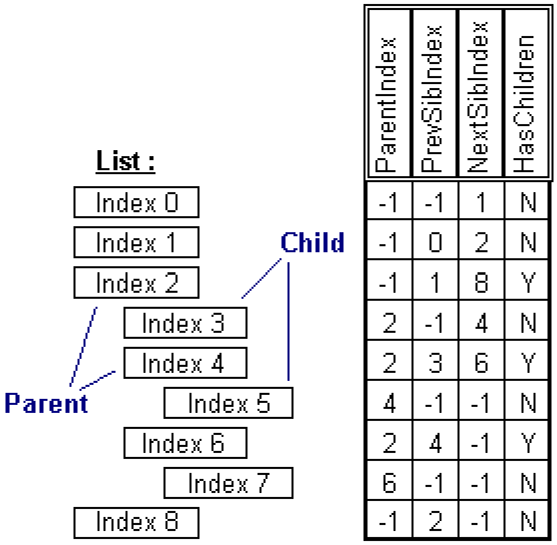
Property Window : No
Des Time : none
Run Time : Read-Write

The ImageZoom Property

See also :

Image Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag

Property Type :

Single, Property Array

Purpose :

Automatically enlarge or shrink images proportions.

Example :

```
'  Make the image display at 50% its original size
img.ImageZoom(Folder Closed) = .5       '   By Name
img.ImageZoom(0) = .5                              '   By index
```

Property Access :

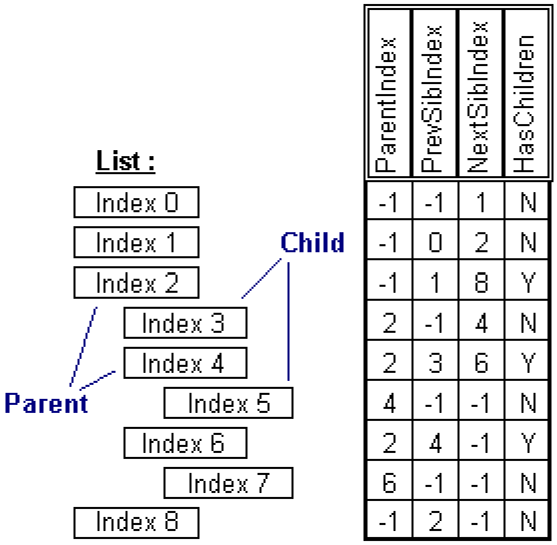
Property Window : No
Des Time : none
Run Time : Read-Write

The IndentWidth Property

See also :

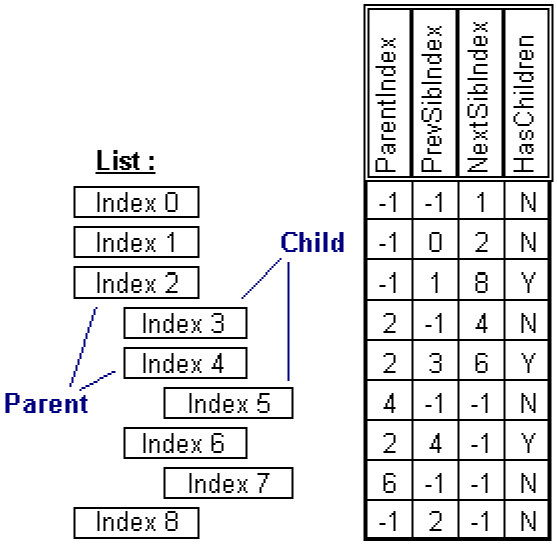
List Display Properties

Applies To :



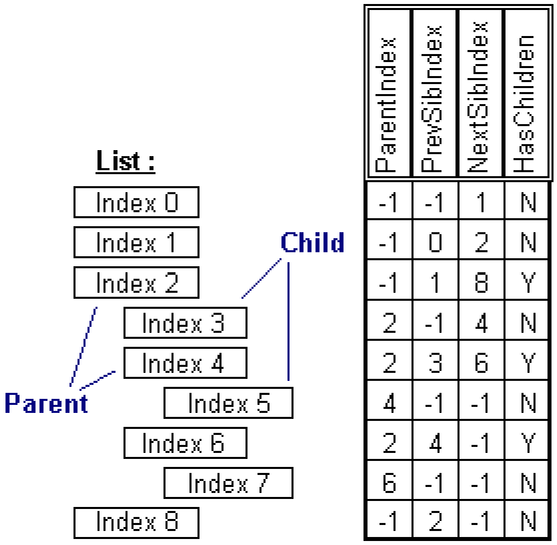
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Integer

Purpose :

the amount of indentation to use when displaying indented items.

Description :

The IndentWidth Property specifies the width (in pixels) to use when Indenting Items that employ the ItemIndent Property. If the ItemIndent Property for an Item is greater than 0 (indicating that this Item should be Indented), the list will display the Item Indented to the Right by (ItemIndent x IndentWidth) pixels.

Example :

```
- Setting Values with Code :
' Set Width and Height to Display 20x20 Bitmaps
List.ItemHeight = 20
List.ImageWidth = 20
' Set IndentWidth to 20 pixels as well
List.IndentWidth = 20

- Getting Values with Code :
Dim I As Integer
I = List.IndentWidth
```

Property Access :

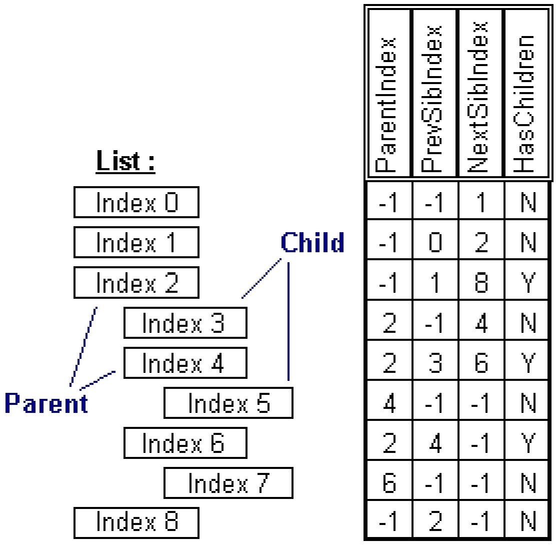
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The ItemData Property

See also :

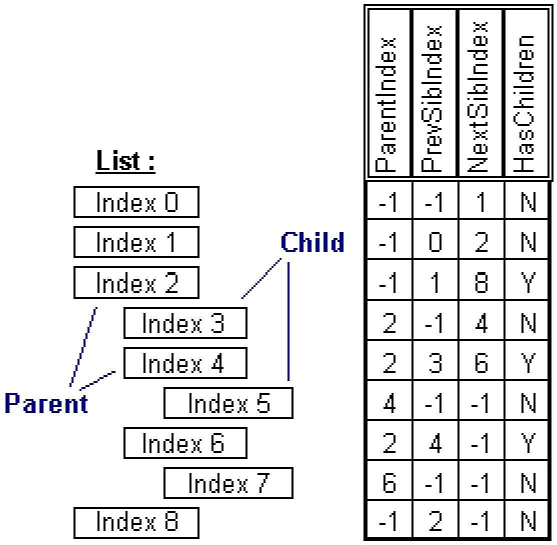
Item Data Properties

Applies To :



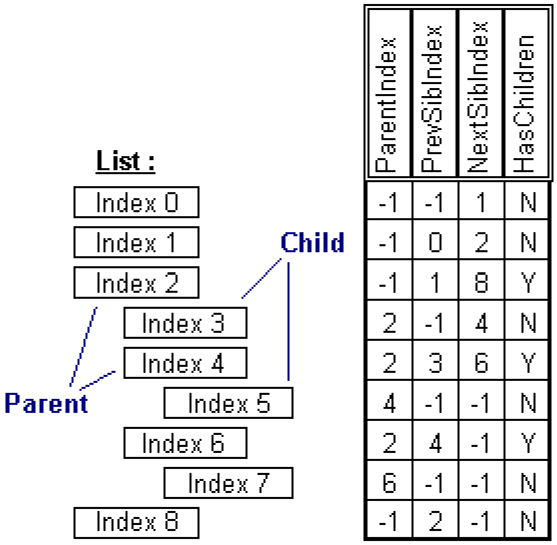
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



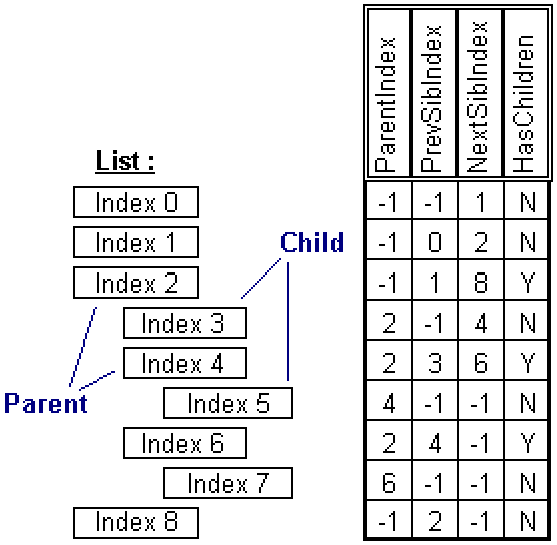
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList



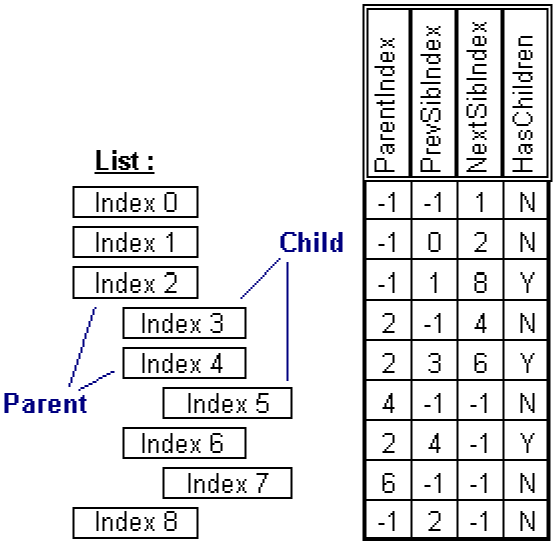
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse



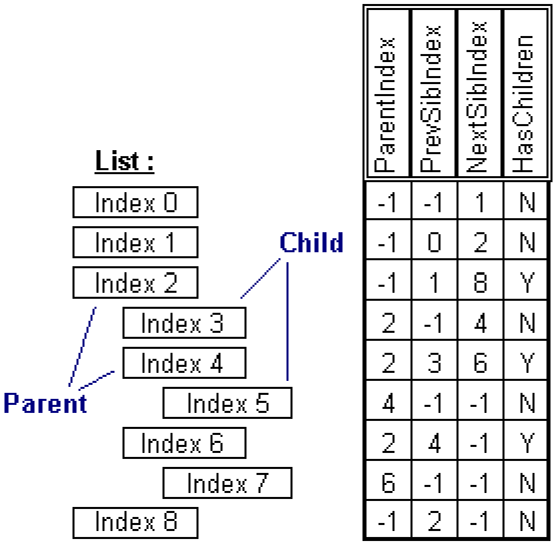
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



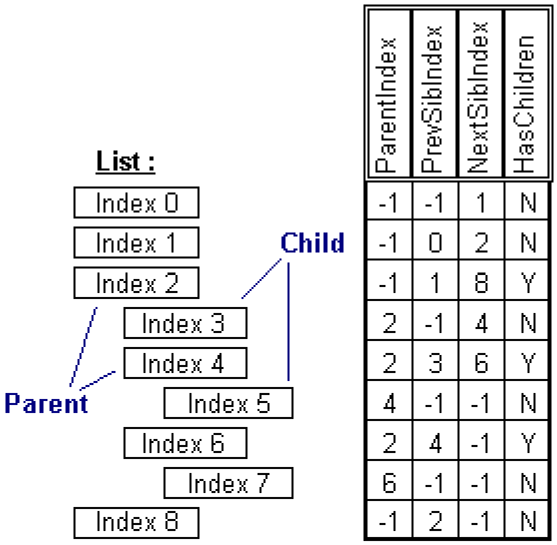
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Long, Property Array

Purpose :

a Long Value that is associated with each Item in the List

Description :

This is Property is useful for associating each item in the list with an array kept in memory or records in a database.

This property is a Property Array.   That is, when setting and getting values with this property, you must also supply an Index (inside the parentheses) in order to tell the List which Value you are talking about.   The property's Index is Zero-Based, meaning that the first Index is 0, the second is 1, etc.,.

Example :

```
- Setting Values with Code :
Dim L As Long
List.ItemData(0) = L

- Getting Values with Code :
Dim L As Long
L = List.ItemData(0)
```

Property Access :

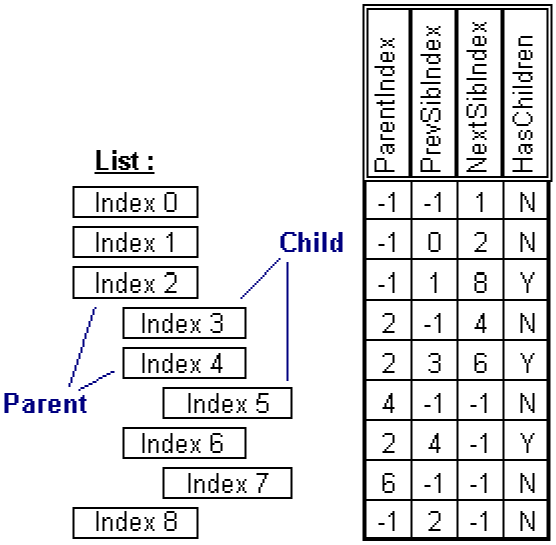
Property Window : No
Des Time : none
Run Time : Read-Write

The ItemEnabled Property

See also :

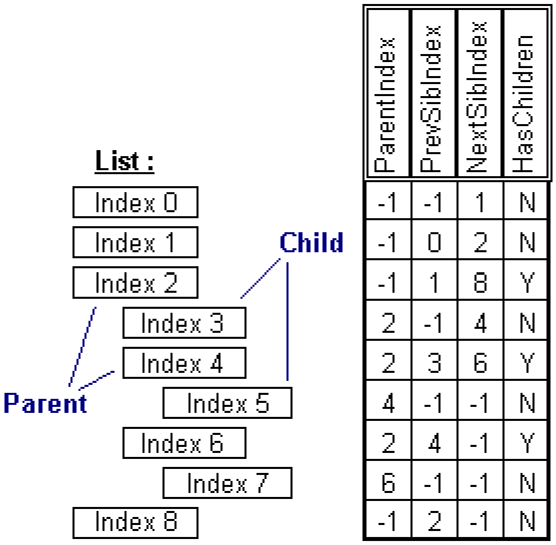
Item Display Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Boolean, Property Array

Purpose :

enable or disable an item

Description :

This property allows you to set individual items as enabled or disabled. By default, all items are initially enabled. By setting an item to be disabled, that item will visible but will not be able to be selected or clicked on.

Example :

```
' Maybe you are filling up a listbox with all the files a person has
' access to. You are using records in a database you have constructed to
' keep track of security.
Dim I as Integer
For I = 0 to (ds.Recordcount-1)
   lst.AddItem ds!FileName
   lst.ItemEnabled(I) = ds!HasAccess
Next I
```

Property Access :

Property Window : No
Des Time : none
Run Time : Read-Write

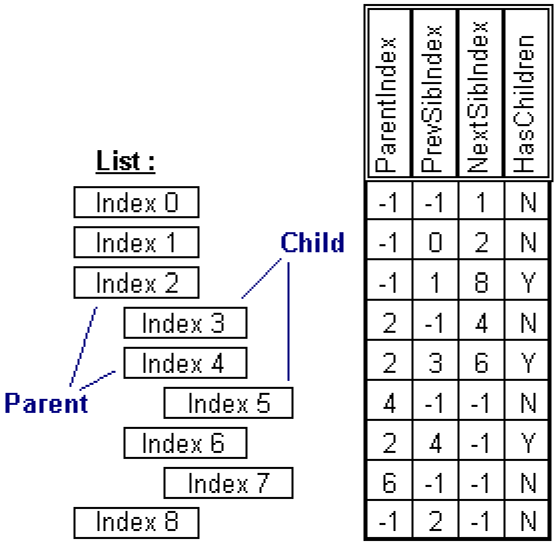
The ItemFlags Property

See also :

Item Display Properties

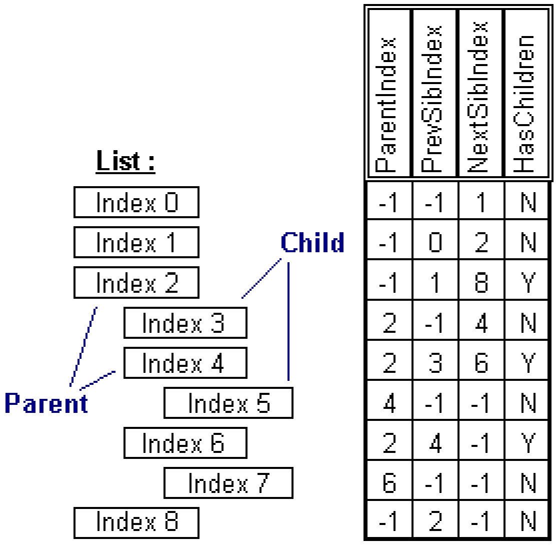
Item Flags String and Long Values

Applies To :



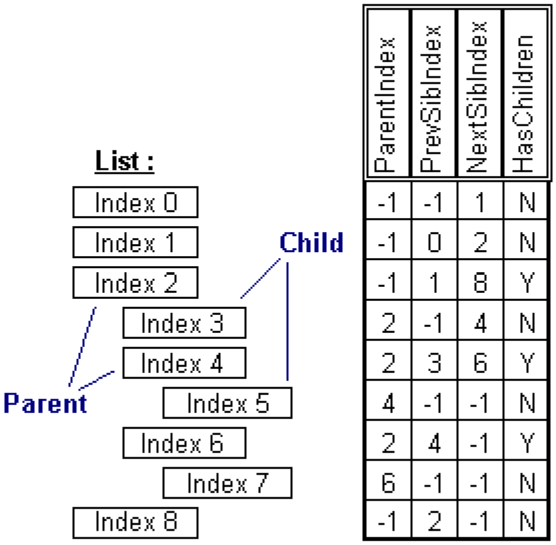
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

String, Property Array

Purpose :

set flags to display special effects within an item

Description :

This property allows you to dynamically alter the display and behaviour of your items in your list easily and efficiently. Using flags (either in string or numeric form) you are able to change the following aspects of item display : Border Display, Item Image placement, Text placement, Text 3D, Shadowing, and Highlighting.

This property accepts and returns a string value; However, the string values are translated to and from long integer values internally. When using Basic, you can use either strings or longs when manipulating the contents of this property, the values will be converted appropriately. Note that when using long integers with this property, it is very much like using the flags in the MsgBox function.

Setting the value of this property is a little different than when setting most other properties. The most powerful way to set the ItemFlags, is to build a string flags to set or unset. Flags must be seperated by a plus '+' or a minus '-'. When preceeded by a plus, the flag will by set within the control. Conversely, a minus preceeding a flag means that flag will be turned off within the control. When using the string method, The new flags will be added to the current flags to arrive at new settings. This is cool because you can customize certain aspects of the Item, and make sure of some of the other settiings at any time. If you use the long integer method, the current flags will be totally replaced by the new flags. The same effect can be forced with the string method by omitting the first plus or minus.

Example :

```
- Setting Values :
Dim s As String
' Keep current settings in the Item but turn on Item's
' Text Shadow and turn off the Highlighting
s = "+TXT_SHADOW-HIL_ALL"
ctrl.ItemFlags(0) = s

- Getting Values :
Dim L As Long
' Get the Control Flags
L = ctrl.ItemFlags
' Check to see if the Text Shadow is set
If ((L And TXT_SHADOW) = TXT_SHADOW) Then
    MsgBox "The Text Shadow is Set!"
End If
```

Property Access :

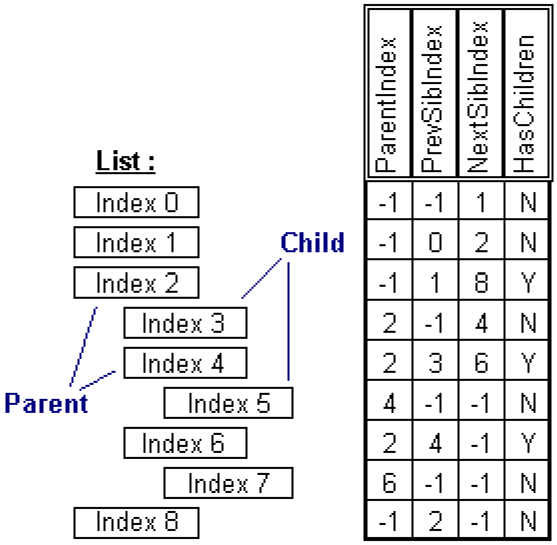
Property Window : No
Des Time : none
Run Time : Read-Write

The ItemHeight Property

See also :

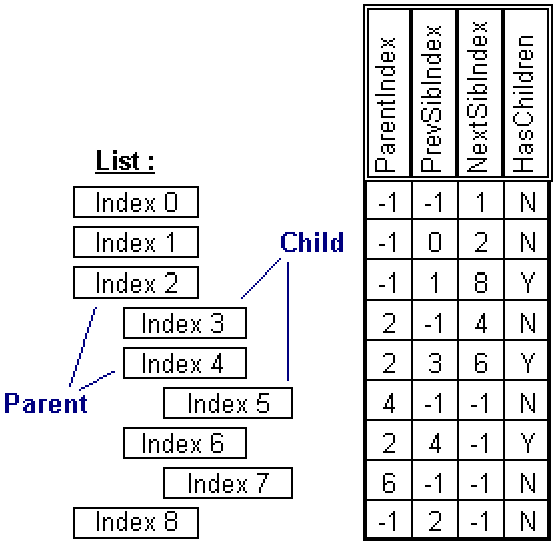
Item Display Properties

Applies To :



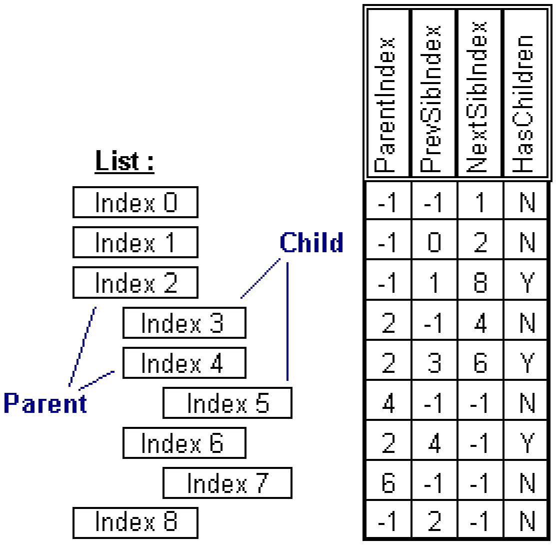
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



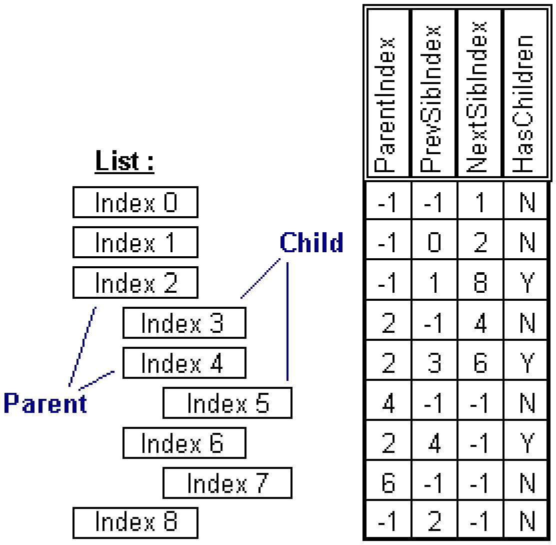
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Integer, Property Array

Purpose :

Height of Item

Description :

Specifies the width of an individual item. If this property is not set, the control will automatically calculate the item's size based on the text, style, and image.
Note that this property must be set in order to enable any word-wrapping set with the ItemFlags property.

Example :

```
see example in ItemTop Property)
```

Property Access :

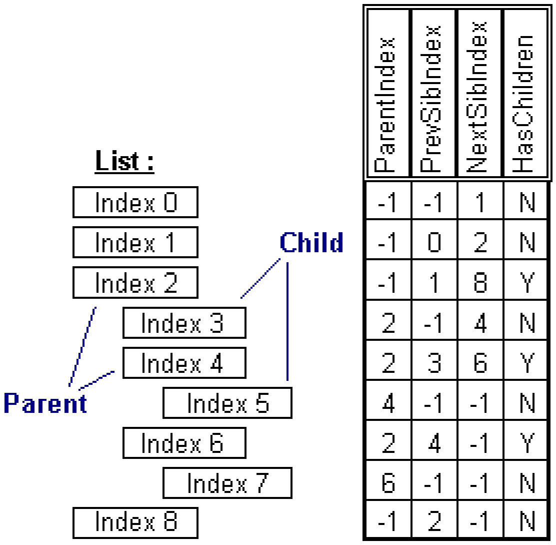
Property Window : No
Des Time : none
Run Time : Read-Write

The ItemImage Property

See also :

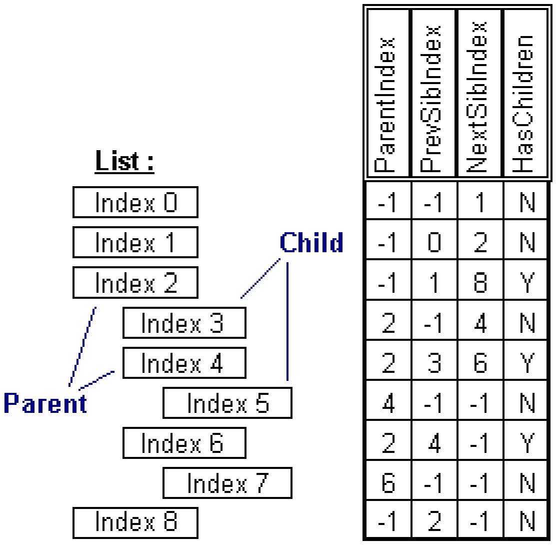
Item Display Properties

Applies To :



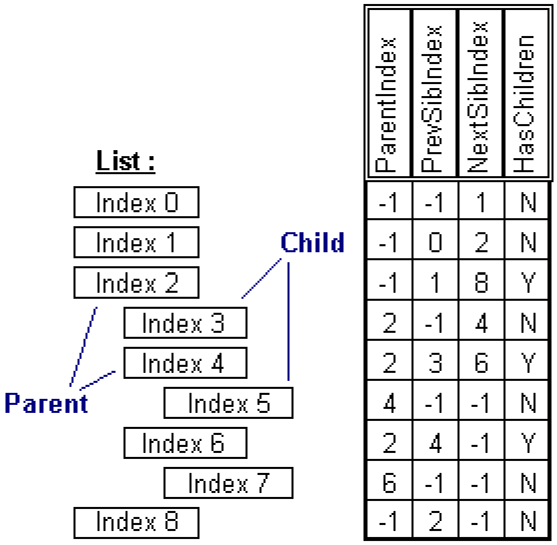
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :
String, Property Array

Purpose :
Image Name or Number for Item

Description :
This property allows you to specify an image that will be displayed with an item. If an item's width and height properties are not set (i.e. they are -1), the item will be automatically sized to include the displayed image.
Note that when using this property, the control which is to display the images must be connected to an ImageBag via the ImageBag property. The value of this property should be the name or index of the image to be used as it occurs in the connected ImageBag.
(This property is not an array for the TSLabel control.)
(For the CTree control, this property must be an integer value identifying one of the nine images available within that control.)

Example :
```
- Setting Values :
lst.ItemImage(1) = LoadPicture(cmDialog1.FileName)
' Example specific to TSTree and TabRas:
lst.ImageBag = ImageBag1.Hwnd
lst.ItemImage(1) = Folder Closed
```

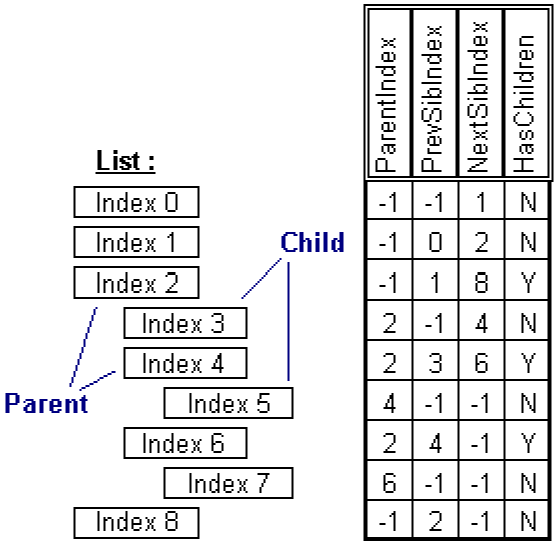
Property Access :
Property Window : No
Des Time : none
Run Time : Read-Write

The ItemIndent Property

See also :

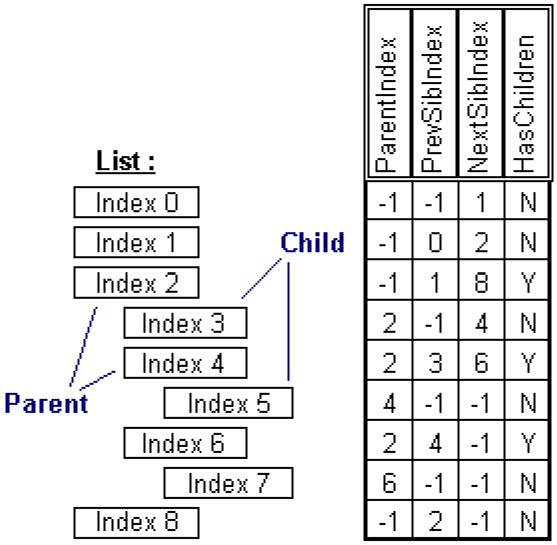
Item Display Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Integer, Property Array

Purpose :

indent level of an item

Description :

The ItemIndent Property specifies the indent level of a particular item.   If an item is indented, the item will be displayed a certain number of pixels from the left-hand side of the list.   This number of pixels is equal to the IndentWidth (number of pixels to indent for each indent level) multiplied by the ItemIndent for each item.

The default ItemIndent for an item is zero (i.e. not indented).   If an item's indent value is greater than zero, then that item's text and any associated ItemImage will be visibly offset to the right of the list (assuming that IndentWidth is greater than zero as well).

The ItemIndent Property opens the door to all the Tree properties included with the list controls. By setting the ItemIndent Property for an item, not only can you adjust the visible display of an item, you can also extract a parent-child relationship between items in the list based upon their order and indentation.

Using the ItemIndent Property to Simulate a Tree Structure
The ItemIndent Property allows you to simulate a tree-like structure with your list. Say you had decided to display a hierarchical structure such as a directory tree or an organizational chart in the list, you may want to indent items in order to visually represent either filenames that appear within a directory, or maybe employees who belong to a particular department.

Since the list control allows you to maintain an indent level for each item in the list with the ItemIndent Property, the list can also provide you with some information concerning the hierarchy of items established through the use of indentation. Such information includes : what item appears as the immediate parent of a group of subitems (ParentIndex), other items that appear at the same indent level and share the same parent item (PrevSibIndex, NextSibIndex), and whether a particular item has any subitems at all (HasChildren).

Example :

```
- Setting Values with Code :
' Add a Department Name
List.AddItem "Accounting"

' List People in that Dept.
List.AddItem "Sally"
List.ItemIndex( List.NewIndex ) = 1
List.AddItem "Bob"
List.ItemIndex( List.NewIndex ) = 1
List.AddItem "Mary"
List.ItemIndex( List.NewIndex ) = 1

- Getting Values with Code :
' See if an Item is a Dept. Name or an Employee Name
Select Case List.ItemIndent(I)
    Case 0: ' Department Name
        MsgBox "This is a Department Name"
    Case 1: ' Employee Name
        MsgBox "This is an Employee Name"
End Select
```

Property Access :

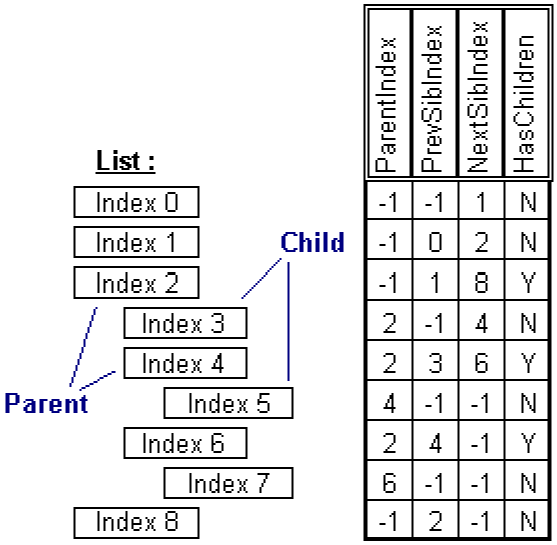
Property Window : No
Des Time : Read-Write
Run Time : Read-Write

The ItemLeft Property

See also :

Item Display Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas

Property Type :

Long, Property Array

Purpose :

Left of Item (in virtual coords)

Description :

In combination with the ItemTop property, this property allows you to set an item's location within the control.
Note that the control must be in FreeForm mode.   That is, no layout flags specified in the ControlFlags property.

Example :

```
 see example in ItemTop Property)
```

Property Access :

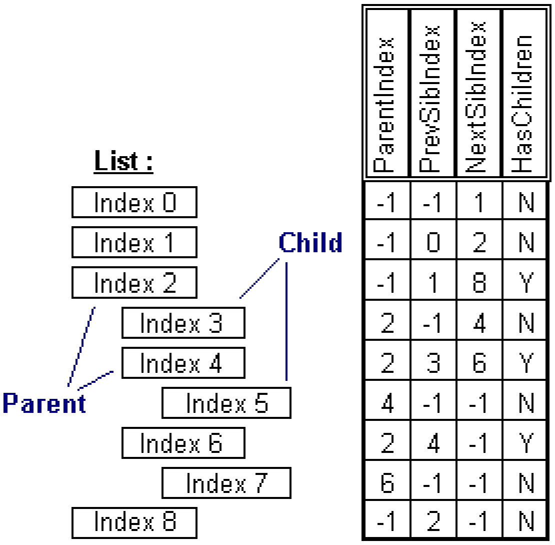
Property Window : No
Des Time : none
Run Time : Read-Write

The ItemSelected Property

See also :

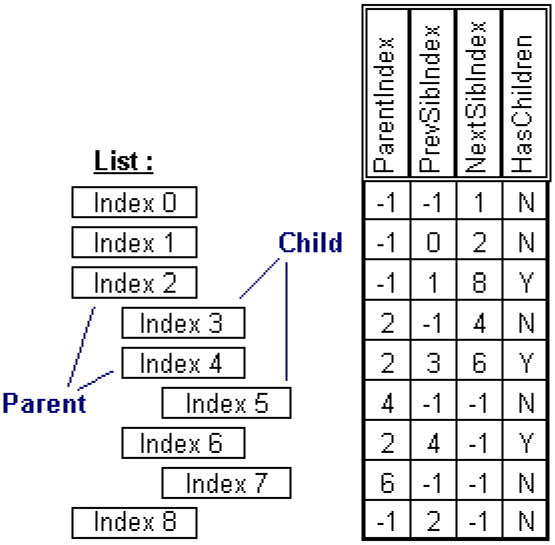
Item Display Properties

Applies To :



List :	ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
Index 0	-1	-1	1	N
Index 1	-1	0	2	N
Index 2	-1	1	8	Y
Index 3	2	-1	4	N
Index 4	2	3	6	Y
Index 5	4	-1	-1	N
Index 6	2	4	-1	Y
Index 7	6	-1	-1	N
Index 8	-1	2	-1	N

TabRas



List :	ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
Index 0	-1	-1	1	N
Index 1	-1	0	2	N
Index 2	-1	1	8	Y
Index 3	2	-1	4	N
Index 4	2	3	6	Y
Index 5	4	-1	-1	N
Index 6	2	4	-1	Y
Index 7	6	-1	-1	N
Index 8	-1	2	-1	N

TSTree

Property Type :

Boolean, Property Array

Purpose :

select or unselect item

Description :

This property allows you to set individual items as selected or unselected.   By default, items are initially unselected.   By setting an item to be selected, it is as if the user had selected the item with the mouse (i.e. the item will highlighted, etc.).   This property is also useful for determining which items the user has selected.   This applies to single and multiple selections.

Example :

```
' You want to perform a certain action only on the items a user has
' selected (multiple select using CTRL or SHIFT keys)
For I = 0 to (lst.ListCount - 1)
   If lst.ItemSelected(I) then
      ' Put your selected-specific code here
       Call SomeSubRoutine()
   End if
Next I
```

Property Access :

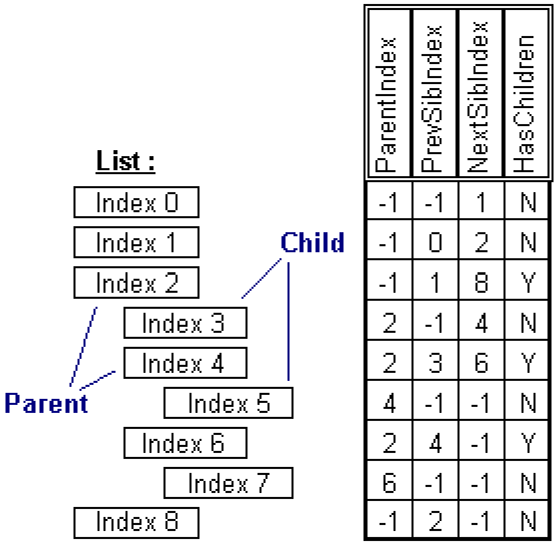
Property Window : No
Des Time : none
Run Time : Read-Write

The ItemSpacing Property

See also :

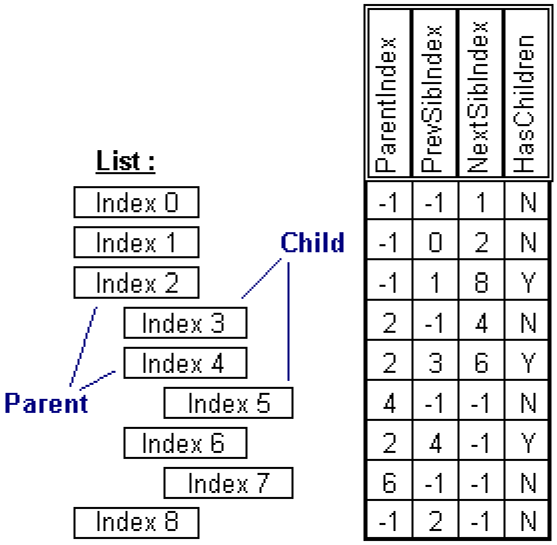
List Display Properties

Applies To :



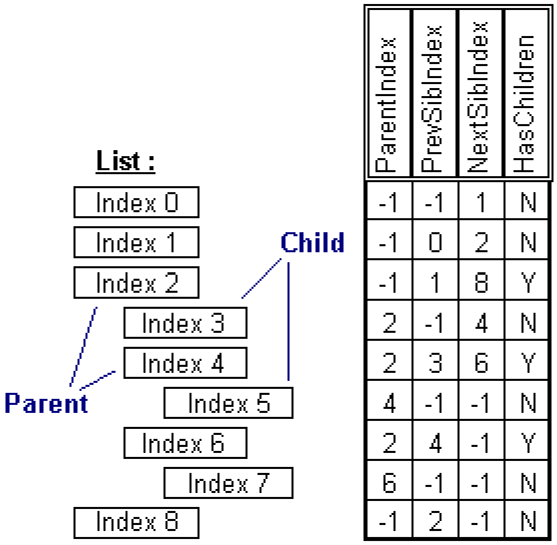
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



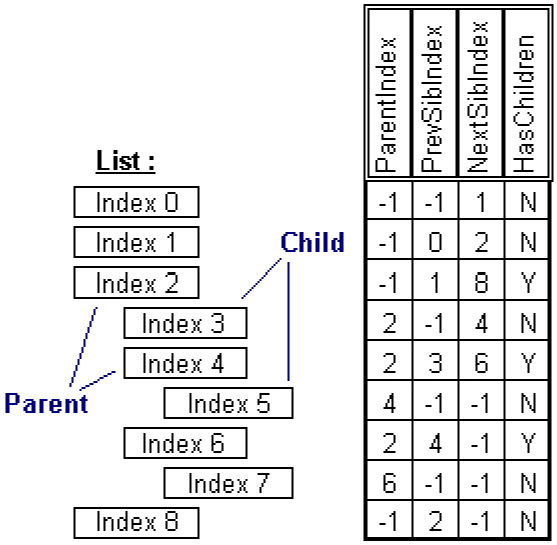
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Integer

Purpose :

spacing to use on the top and left of items

Description :

This property specifies the number of pixels to use for spacing between items.   The ItemSpacing is applied to the Top and Left of each Item.

Property Access :

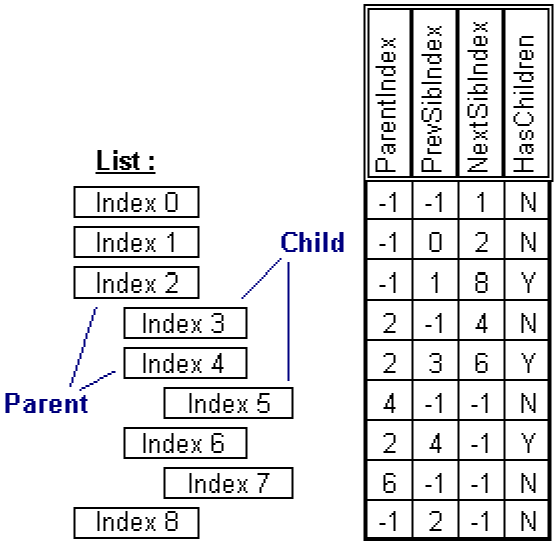
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The ItemStateImage Property

See also :

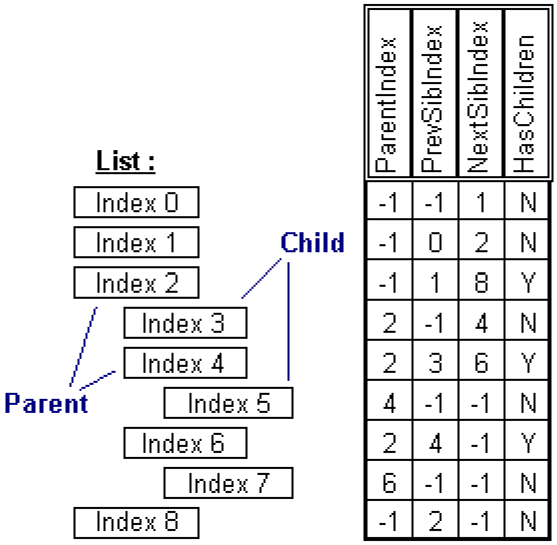
Item Display Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

String, Property Array

Purpose :

an image displayed outside and to the left of an item

Description :

This property allows you to specify an image that will be displayed outside and to the left of an item. This image is best used in combination with the DefStateImage property to automatically display certain images based upon an item's state (current, selected, parent, etc.). This property can also be set for each item to display custom state information.

Note that when using this property, the control which is to display the images must be connected to an ImageBag via the ImageBag property. The value of this property should be the name or index of the image to be used as it occurs in the connected

ImageBag.

Example :

```
' In the ImageBag you have an image with the ImageName Selected
lst.ImageBag = ImageBag1.Hwnd
lst.ItemStateImage(1) = Selected
```

Property Access :

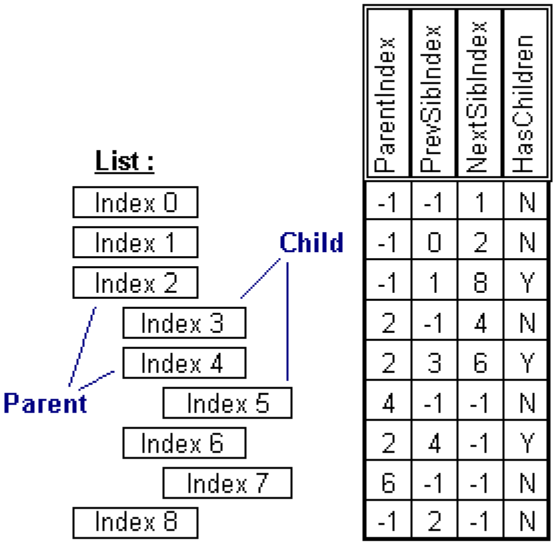
Property Window : No
Des Time : none
Run Time : Read-Write

The ItemString Property

See also :

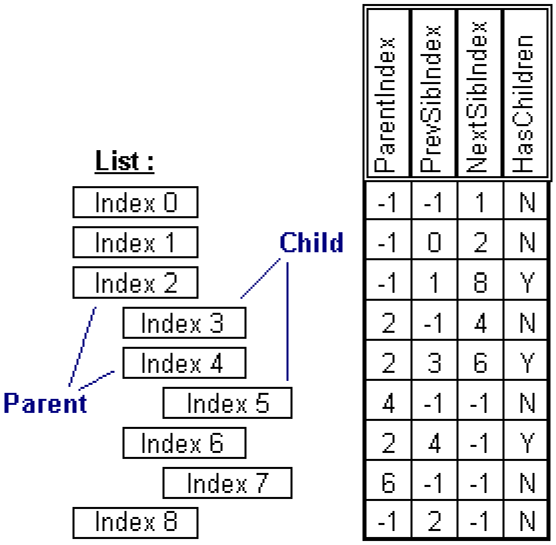
Item Data Properties

Applies To :



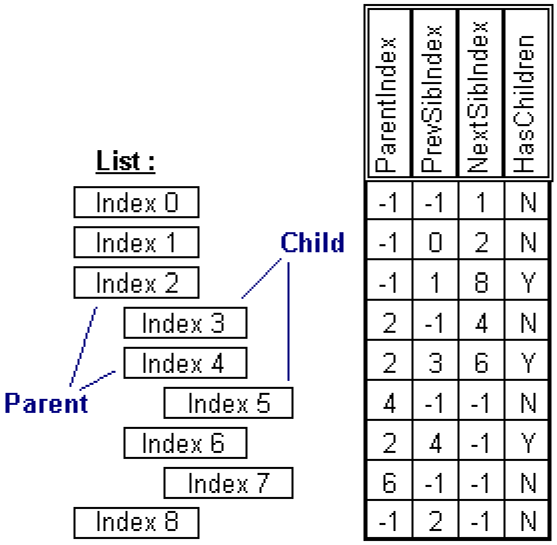
List :	ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
Index 0	-1	-1	1	N
Index 1	-1	0	2	N
Index 2	-1	1	8	Y
Index 3	2	-1	4	N
Index 4	2	3	6	Y
Index 5	4	-1	-1	N
Index 6	2	4	-1	Y
Index 7	6	-1	-1	N
Index 8	-1	2	-1	N

CTree



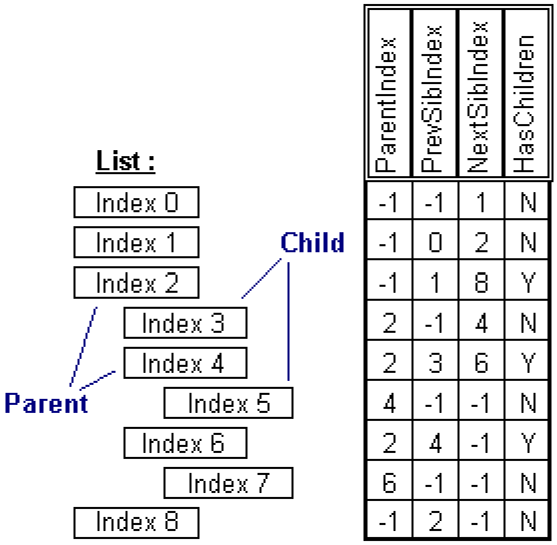
List :	ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
Index 0	-1	-1	1	N
Index 1	-1	0	2	N
Index 2	-1	1	8	Y
Index 3	2	-1	4	N
Index 4	2	3	6	Y
Index 5	4	-1	-1	N
Index 6	2	4	-1	Y
Index 7	6	-1	-1	N
Index 8	-1	2	-1	N

EnhList



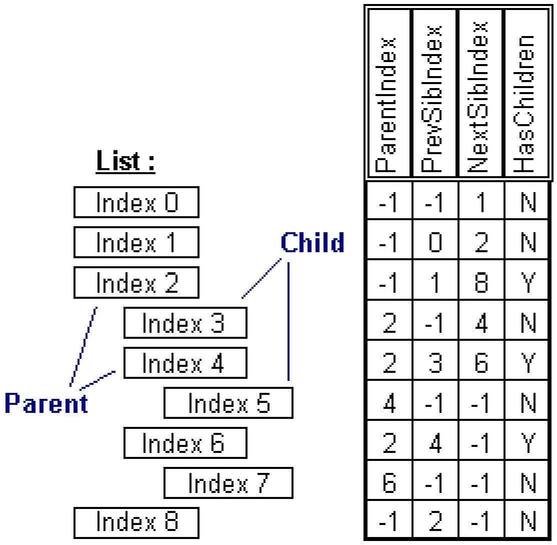
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse



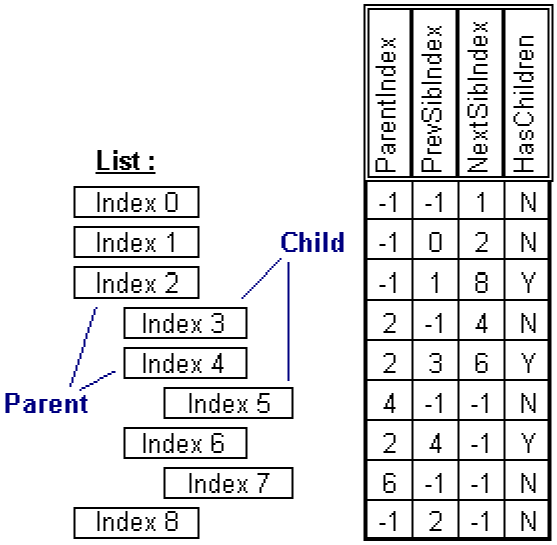
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



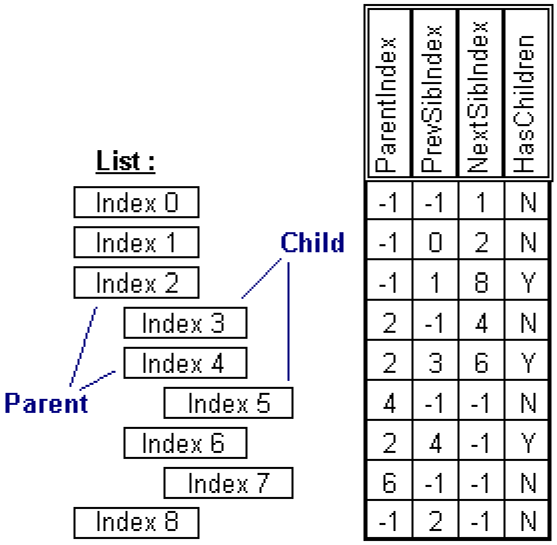
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

String, Property Array

Purpose :

a String Value that is associated with each Item (alias for ExtraString)

Description :

This property, similar to the ItemData property, can be used to associate each item in a List with extra data stored elsewhere (for example, the ItemString may hold a BookMark string for each record in a table). The ItemString may also be used to store your data, with the ItemText property displaying only what you want the user to see.

This property is a Property Array. That is, when setting and getting values with this property, you must also supply an Index (inside the parentheses) in order to tell the List which Item you are talking about. The property's Index is Zero-Based, meaning that the first Index is 0, the second is 1, etc.,.

Example :

```
- Setting Values with Code :
Dim S As String
S = "This Text is not going to be Displayed"
List.ExtraString( 0 ) = S

- Getting Values with Code :
Dim S As String
S = List.ExtraString( 0 )
```

Property Access :

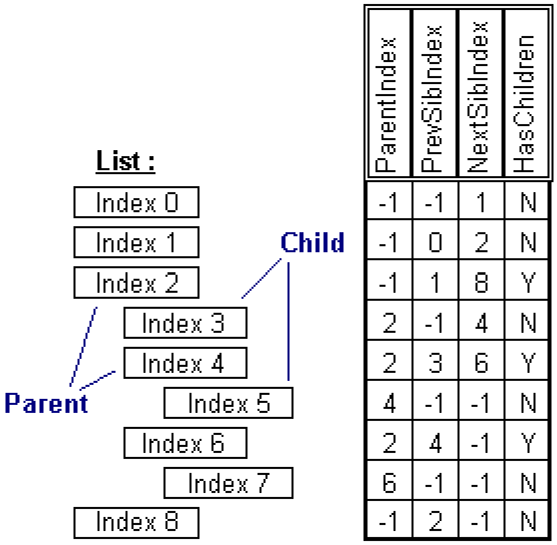
Property Window : No
Des Time : Read-Write
Run Time : Read-Write

The ItemStyle Property

See also :

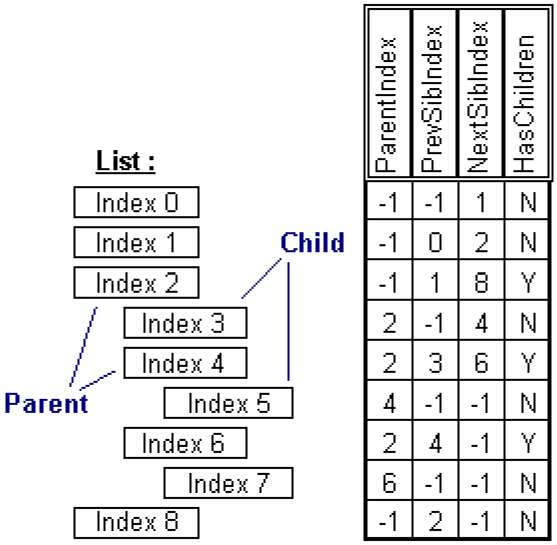
Item Display Properties

Applies To :



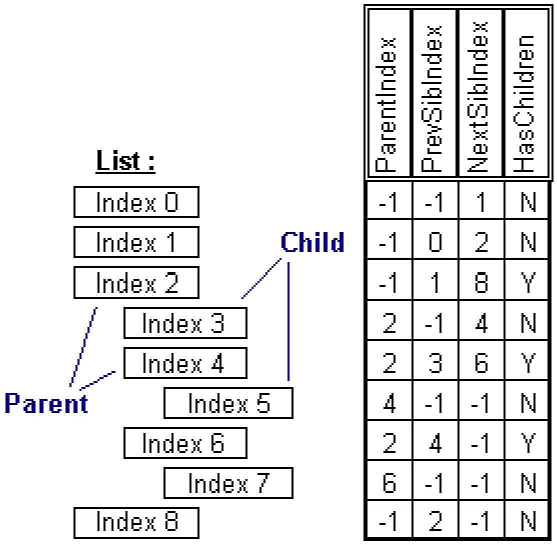
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :
String, Property Array

Purpose :
Style Name or Number for Item

Description :
This property allows you to specify a style with which to display an item.   An item's style may dictate that item's size, font, color, border, and also contributes to the item's ItemFlags property.

Note that when using this property, the control which is to display the images must be connected to a StyleBag via the StyleBag property.   The value of this property should be the name or index of the style to be used as it occurs in the connected StyleBag. (This property is not an array for the TSLabel control.)

Example :
```
' In the StyleBag you have an Style with the StyleName "Selected"
lst.StyleBag = StlyeBag1.Hwnd
lst.ItemStyle(1) = "Selected"
```

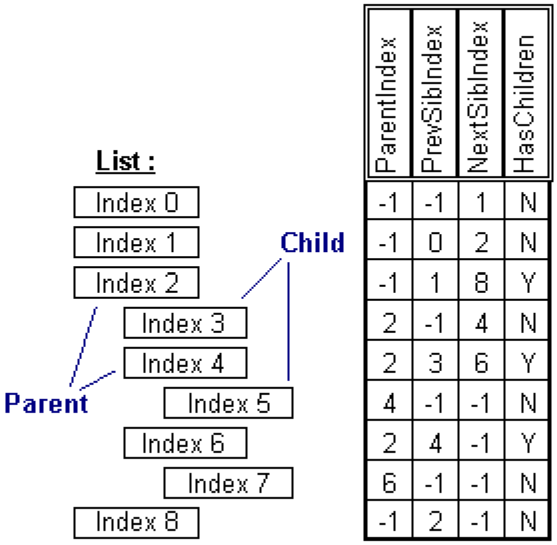
Property Access :
Property Window : No
Des Time : none
Run Time : Read-Write

The ItemText Property

See also :

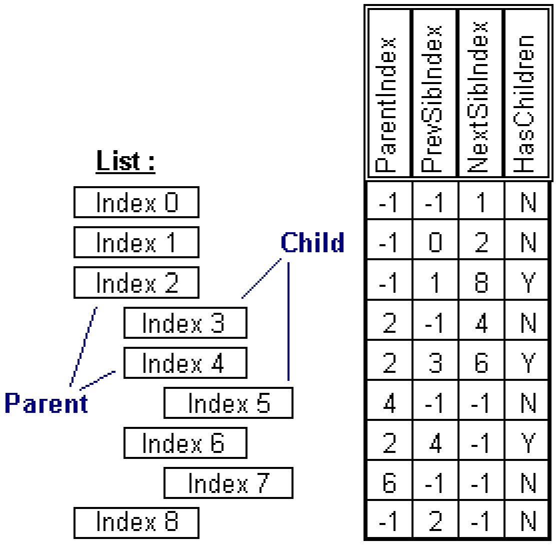
Item Data Properties

Applies To :



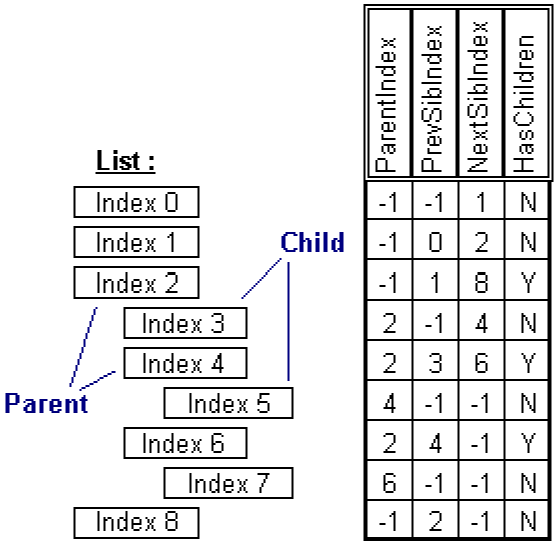
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :
String, Property Array

Purpose :
formatted text display (alias for List Property)

Description :
The ItemText Property is the text that is displayed as each item in the List.

This property is a Property Array.   That is, when setting and getting values with this property, you must also supply an Index (inside the parentheses) in order to tell the List which Value you are talking about.   The property's Index is Zero-Based, meaning that the first Index is 0, the second is 1, etc.,.

Example :
```
- Setting Values with Code :
Dim S As String
S = "This is what is Displayed as the First Item"
List.ItemText(0) = S

- Getting Values with Code :
Dim S As String
S = List.ItemText(0)
```

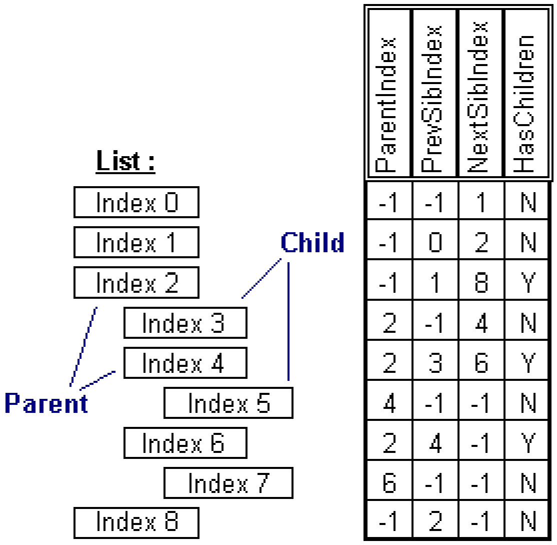
Property Access :
Property Window : No
Des Time : none
Run Time : Read-Write

The ItemTop Property

See also :

Item Display Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas

Property Type :

Long, Property Array

Purpose :

Top of Item (in virtual coords)

Description :

In combination with the ItemLeft property, this property allows you to set an item's location within the control.

Note that the control must be in FreeForm mode.   That is, no layout flags specified in the ControlFlags property.

Example :

```
- Setting Values with Code :
' Remove all Layout flags
TabRas1.ControlFlags = "-LAY_ALL"

' Add an Item
TabRas1.AddItem "This is an Item"

' Set the Item's Position
TabRas1.ItemTop(TabRas1.NewIndex) = 10
TabRas1.ItemLeft(TabRas1.NewIndex) = 15

' Set the Item's Size (optional - the control will size it if omtted)
TabRas1.ItemWidth(TabRas1.NewIndex) = 100
TabRas1.ItemHeight(TabRas1.NewIndex) = 25
```

Property Access :

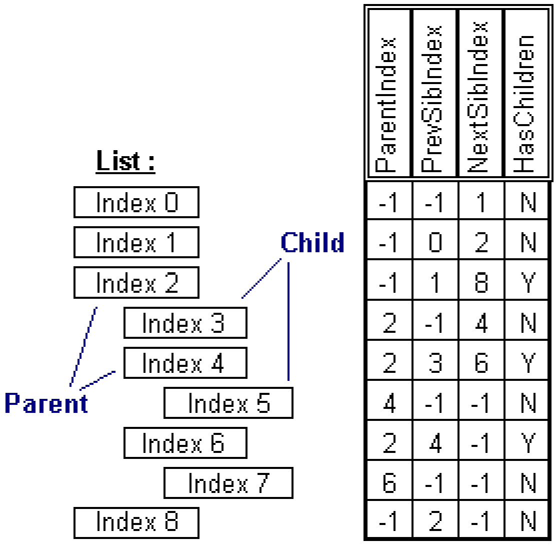
Property Window : No
Des Time : none
Run Time : Read-Write

The ItemVisible Property

See also :

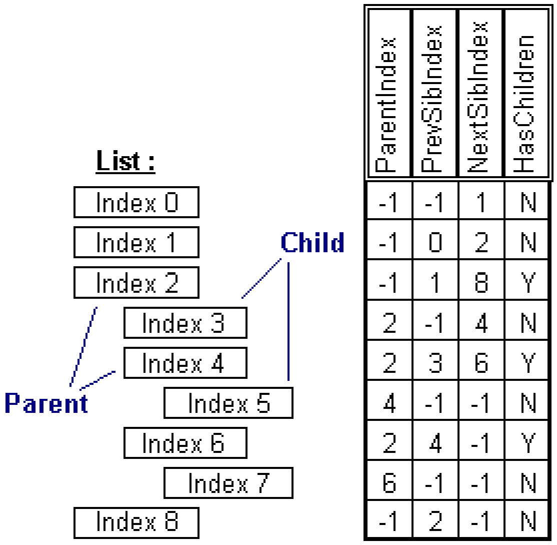
Item Display Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Boolean, Property Array

Purpose :

set an item's visibility on or off

Description :

This property allows you to make individual items visible or invisible. By default, an item is initially visible. When make an item invisible, any currently visible children are also set invisible as well.

Example :

```
' Maybe you are filling up a listbox with all the files a person has
' access to. You are using records in a database you have constructed to
' keep track of security.
Dim I as Integer

For I = 0 to (ds.Recordcount-1)
   lst.AddItem ds!FileName
   lst.ItemVisible(I) = ds!HasAccess
Next I
```

Property Access :

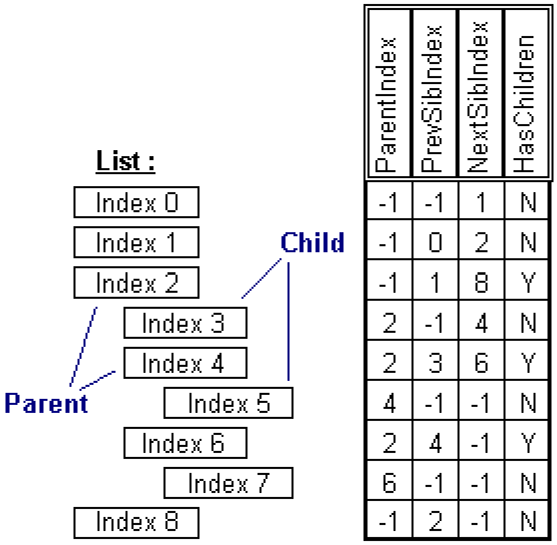
Property Window : No
Des Time : none
Run Time : Read-Write

The ItemWidth Property

See also :

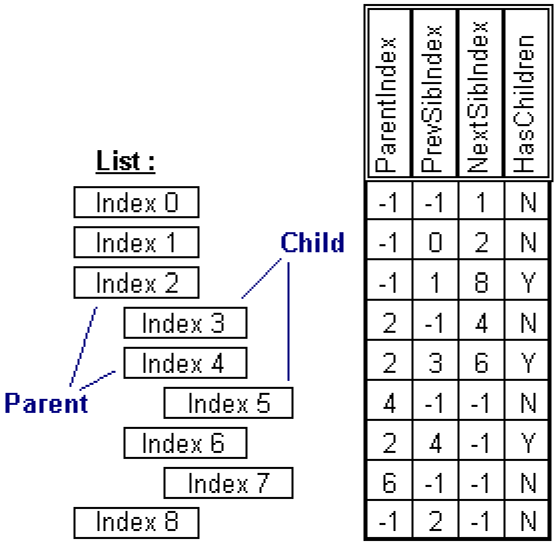
Item Display Properties

Applies To :



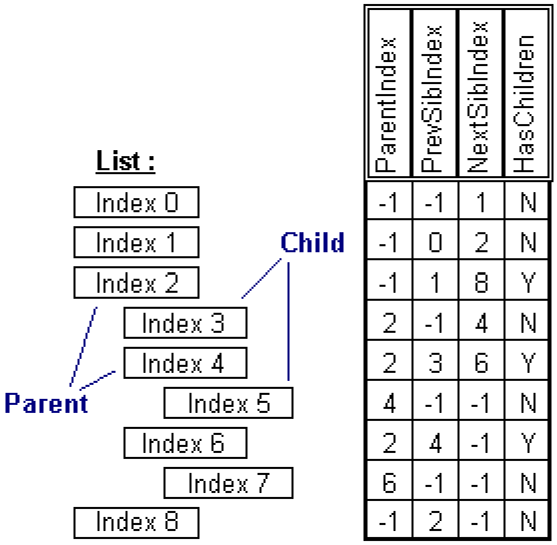
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Integer, Property Array

Purpose :

Width of Item

Description :

Specifies the width of an individual item.   If this property is not set, the control will automatically calculate the item's size based on the text, style, and image.
Note that this property must be set in order to enable any word-wrapping set with the ItemFlags property.

Example :

```
 see example in ItemTop Property)
```

Property Access :

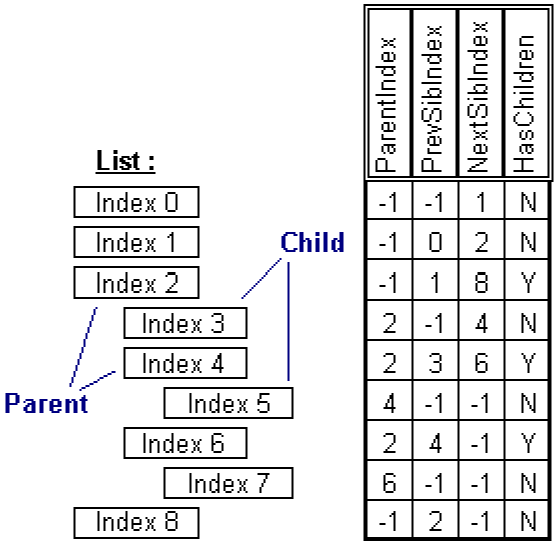
Property Window : No
Des Time : none
Run Time : Read-Write

The List Property

See also :

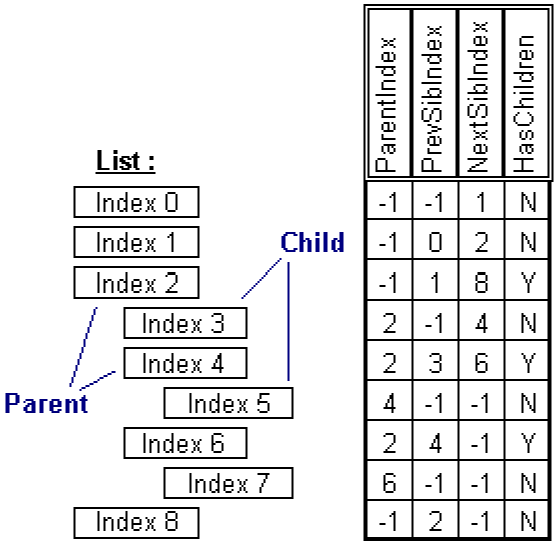
Item Data Properties

Applies To :



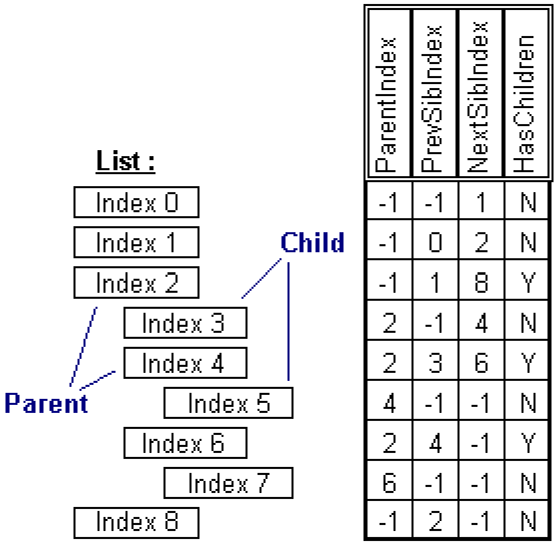
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



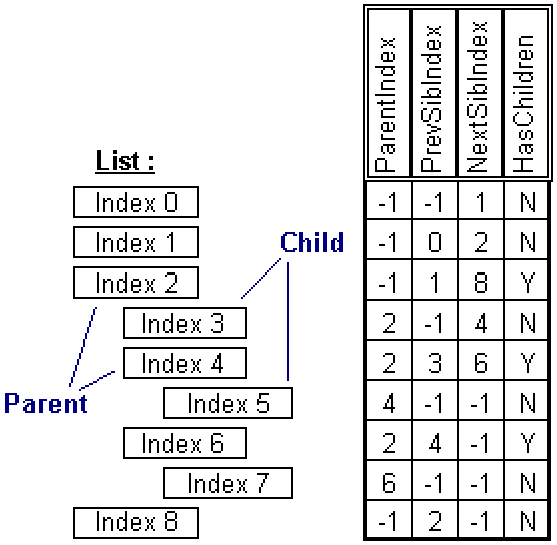
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList



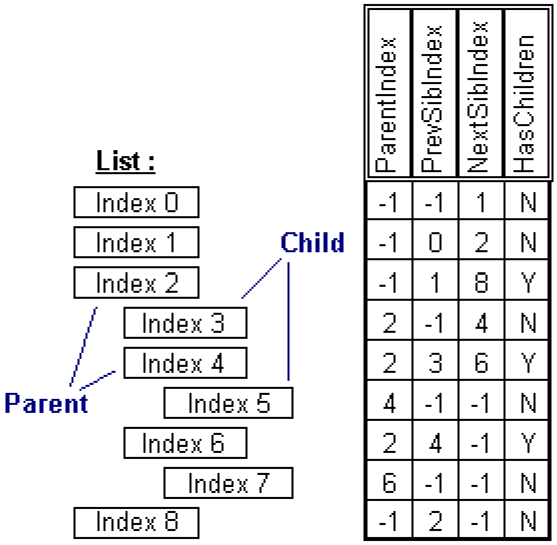
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



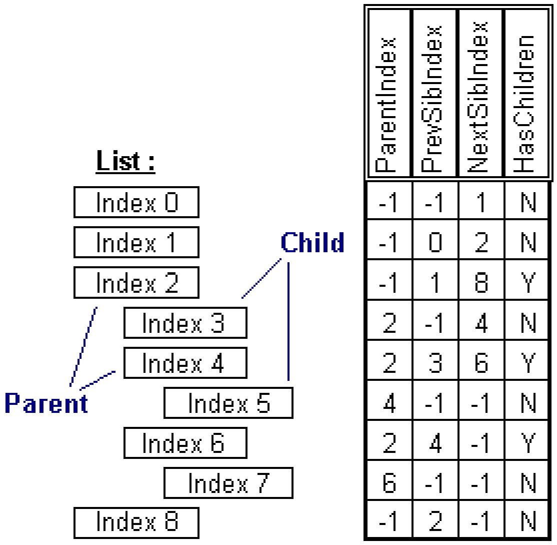
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse



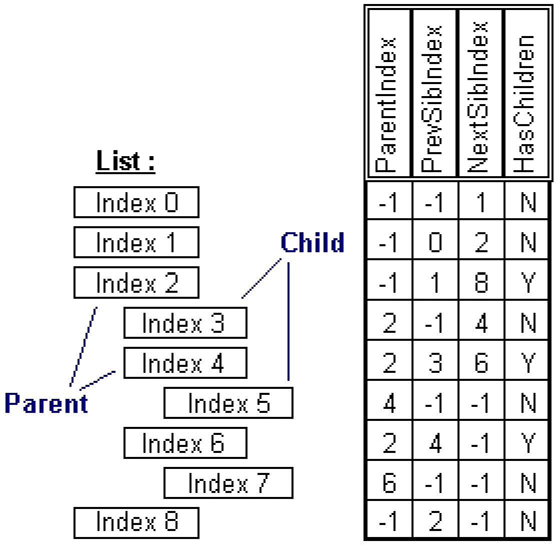
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



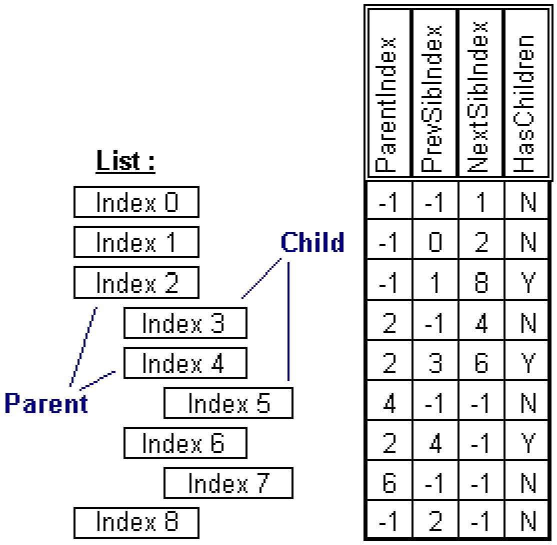
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

String, Property Array

Purpose :

item's displayed text and main string storage

Description :

The List Property is the List Text that is displayed as each item in the List.
This property is a Property Array. That is, when setting and getting values with this property, you must also supply an Index (inside the parentheses) in order to tell the List which Value you are talking about. The property's Index is Zero-Based, meaning that the first Index is 0, the second is 1, etc.,.

This property is probably the most important of the List properties. It controls what is displayed to the user in the List.

Example :

```
- Setting Values with Code :
Dim S As String
S = 'This is what is Displayed as the First Item"
List.List(0) = S
```

```
- Getting Values with Code :
Dim S As String
S = List.List(0)
```

Property Access :

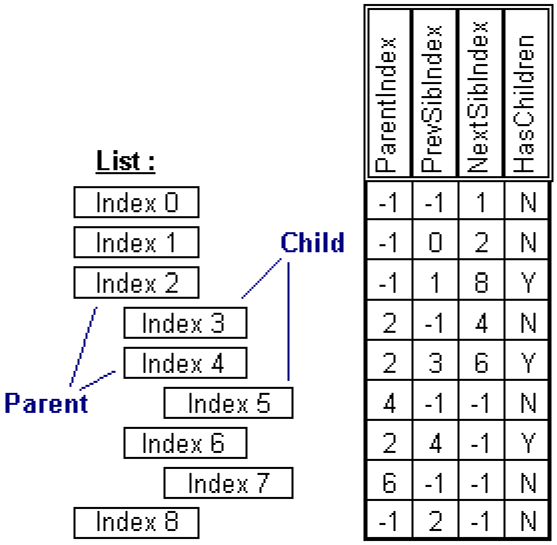
Property Window : No
Des Time : none
Run Time : Read-Write

The ListCount Property

See also :

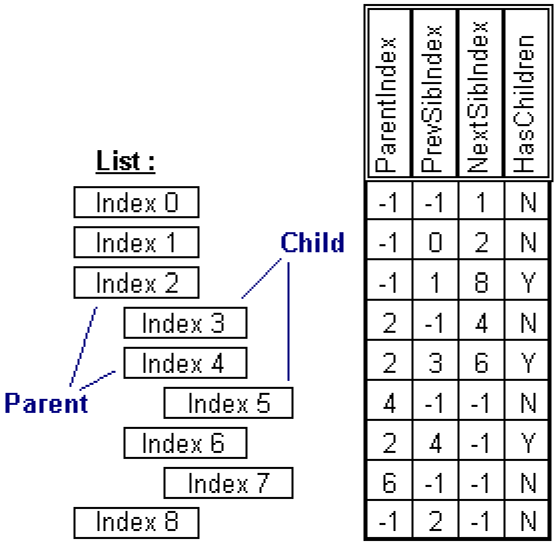
List Data Properties

Applies To :



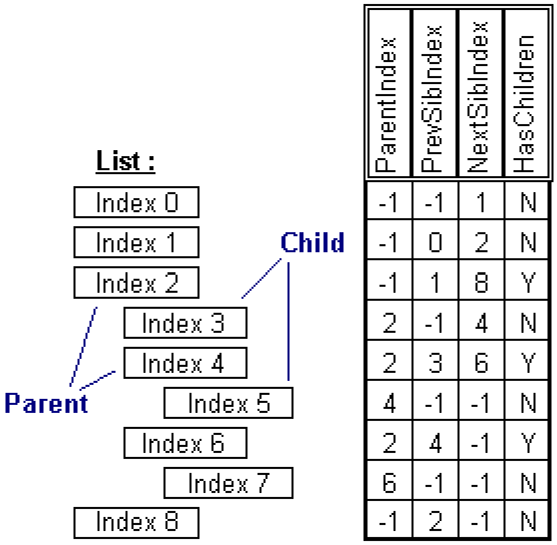
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



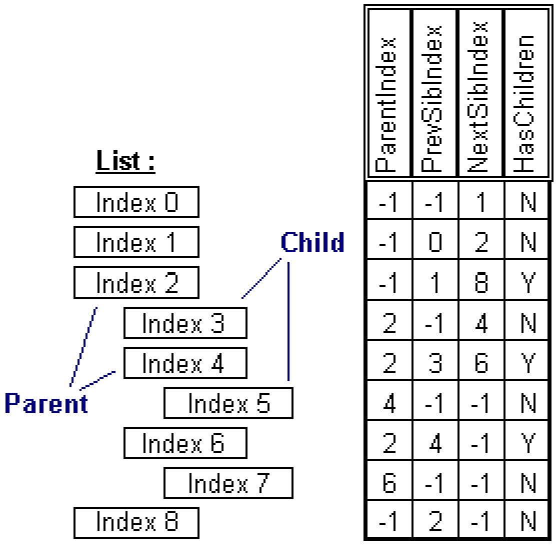
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList



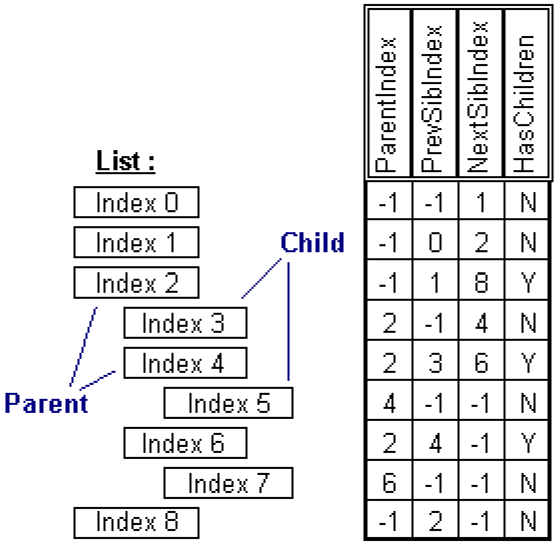
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



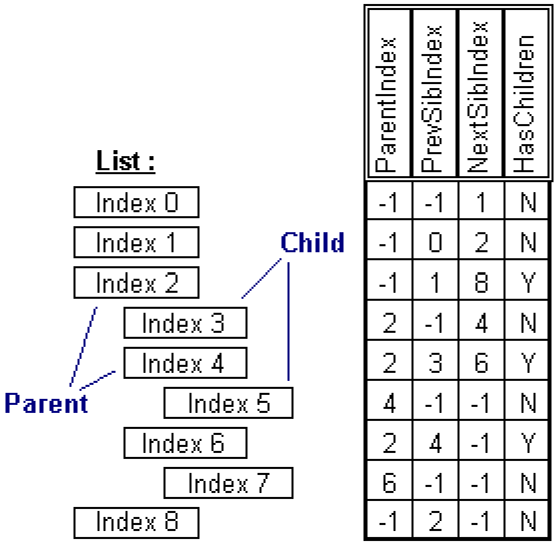
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse



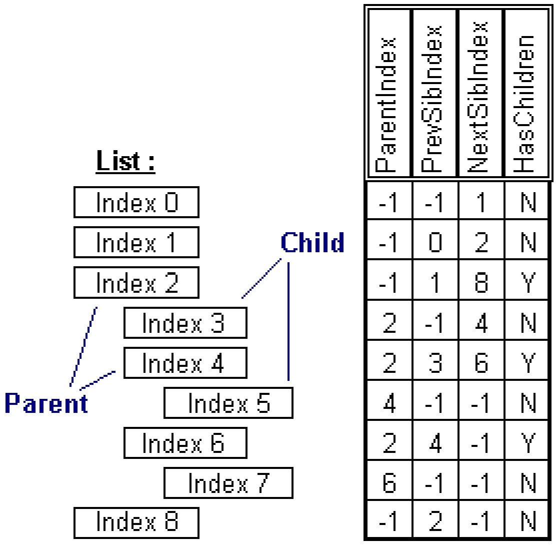
ParentIndex	PrevSiblndex	NextSiblndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



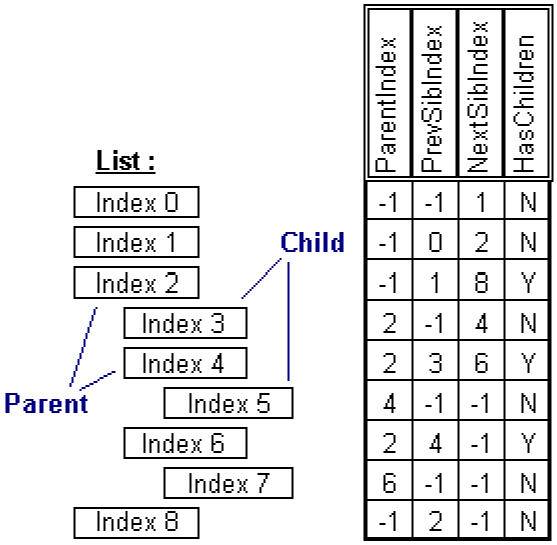
ParentIndex	PrevSiblndex	NextSiblndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Long

Purpose :

indicates the total number of Items that have been added to a List

Description :

This is useful when a program needs to know the number if items in a List or to loop through all of the Items in a List.

Example :

```
- Setting Values with Code :
This Property is Read-Only at RunTime

- Getting Values with Code :
Dim L as Long
L = List.ListCount
```

Property Access :

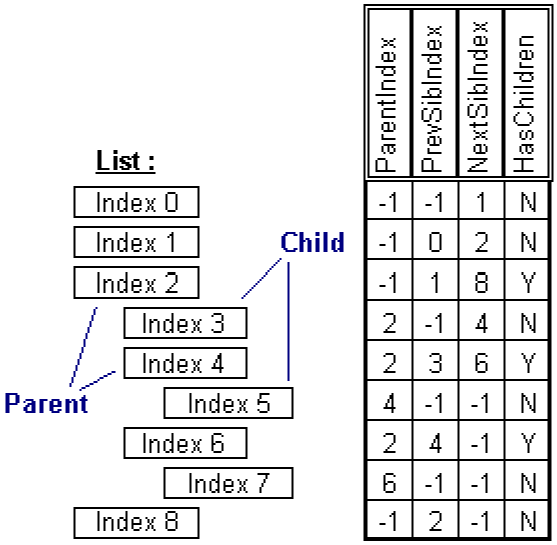
Property Window : No
Des Time : none
Run Time : Read-Only

The ListDir Property

See also :

File/Dir Viewing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList

Property Type :

String

Purpose :

the path (with wildcards) to display

Description :

The ListDir Property is used to obtain a listing of the contents of a specified path, much the same way the DOS Dir command works. Wildcard characters are included to specify certain files as well. The result of setting this property, is that the list will clear itself and list all the matching directory entries (subdirectories and files) found for the ListDir string. All entries are added in alphabetical order. When filenames are added to the list, the ExtraString Property is also filled with the file's attributes, date, time, and size (directory names will have nothing in the ExtraString property) as follows ("_" denotes a space) :
RASH_MM\DD\YY_HH:MM:SS_size
For example, a filename in the list which is Read-Only, has a date of 1/5/95, a time of 3:14:00 pm, and a size of 11,096 bytes would have an ExtraString as follows :
R____01/05/95_15:14:00_11096

Note: This property combines the functionality of the standard Visual Basic Directory and File listboxes. It also provides more detailed information about the files listed in the listbox.

Example :

```
- Setting Values with Code :
' List All Files in C:\
List.ListDir = "C:\."
' List All Text Files in C:\
List.ListDir = "C:\*.TXT"

- Getting Values with Code :
Dim S As String
S = List.ListDir
```

Property Access :

Property Window : No
Des Time : none

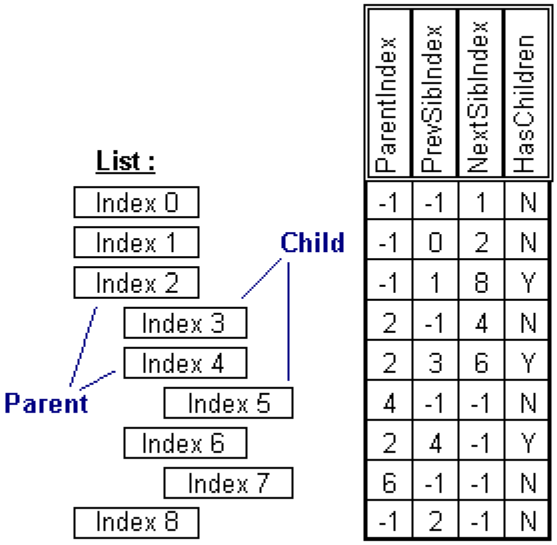
Run Time : Read-Write

The ListIndex Property

See also :

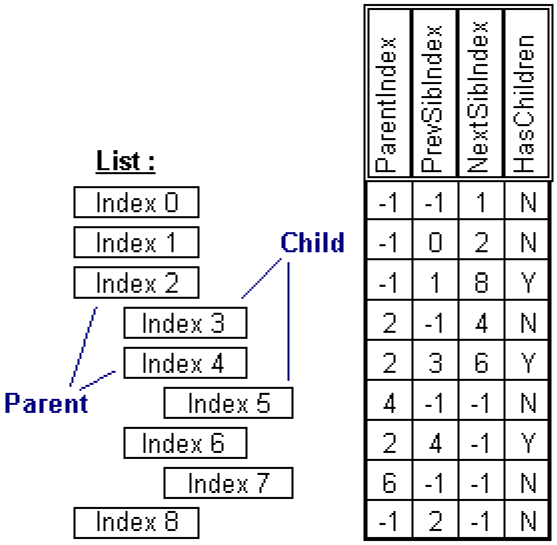
List Data Properties

Applies To :



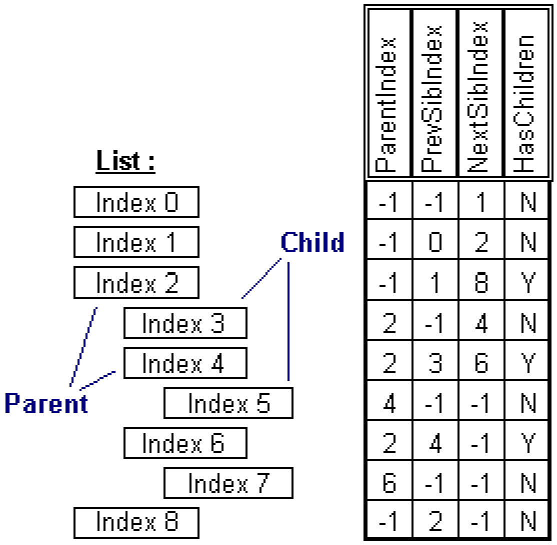
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



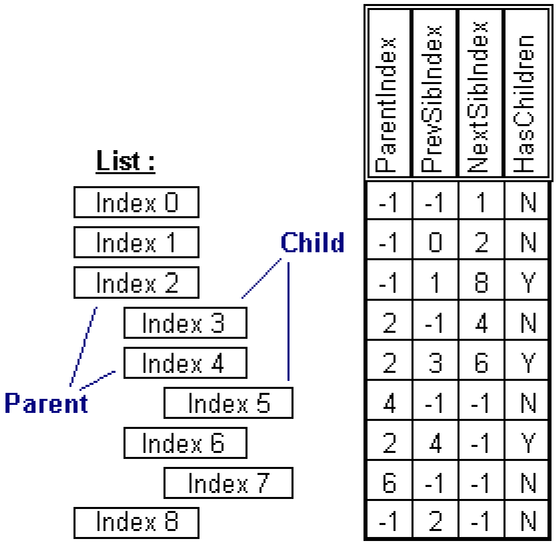
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList



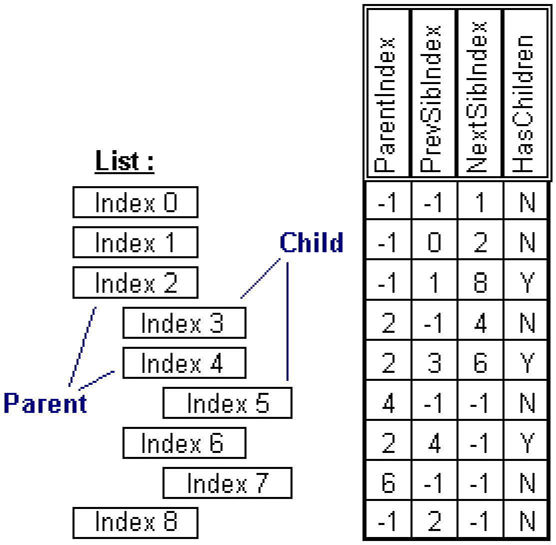
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



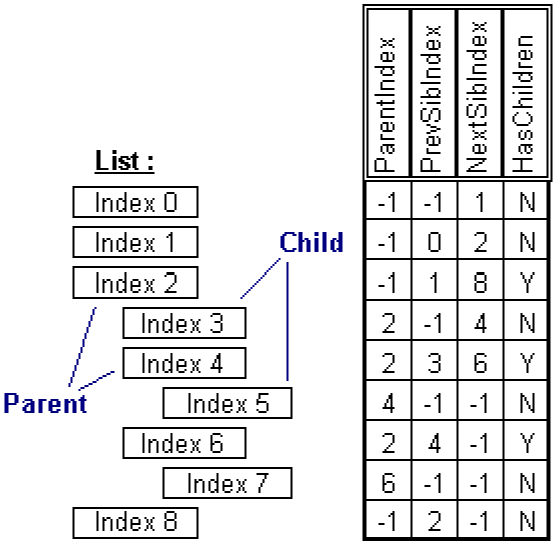
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse



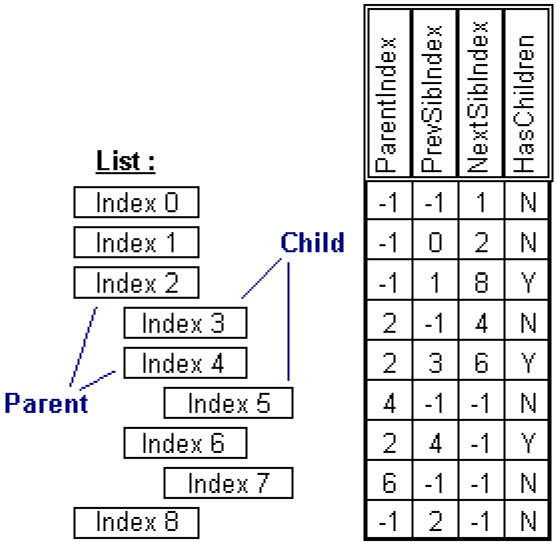
ParentIndex	PrevSiblIndex	NextSiblIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



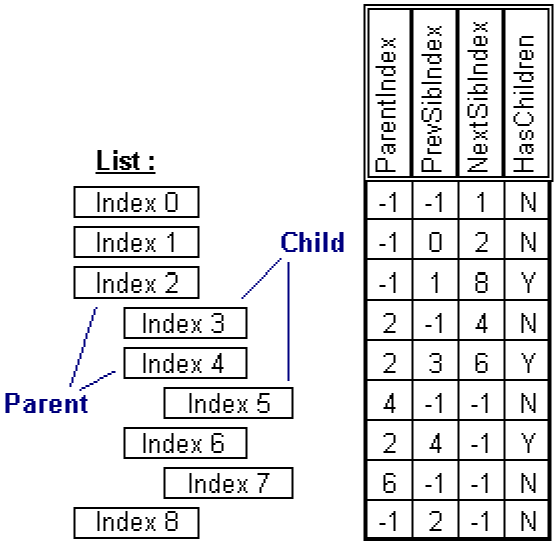
ParentIndex	PrevSiblIndex	NextSiblIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Long

Purpose :

Use the ListIndex property to determine which Item has been clicked on by a user, or to programmatically select an Item.

Description :

The ListIndex Property determines the currently selected/highlighted Item, if any, in the List. If no Item is currently selected in the List, then the value of the ListIndex will be -1.

Note: Setting this value at RunTime will fire a Click and IndexChange Event for the control (as in the standard ListBox control), but not an ItemClick Event.

Example :

```
- Setting Values with Code :
' To Deselect any Selected Item
List.ListIndex = -1
' To Select the First Item
List.ListIndex = 0
' To Select the Last Item
List.ListIndex = (List.ListCount - 1)
```

```
- Getting Values with Code :
Dim I as Integer
I = List.ListIndex
```

Property Access :

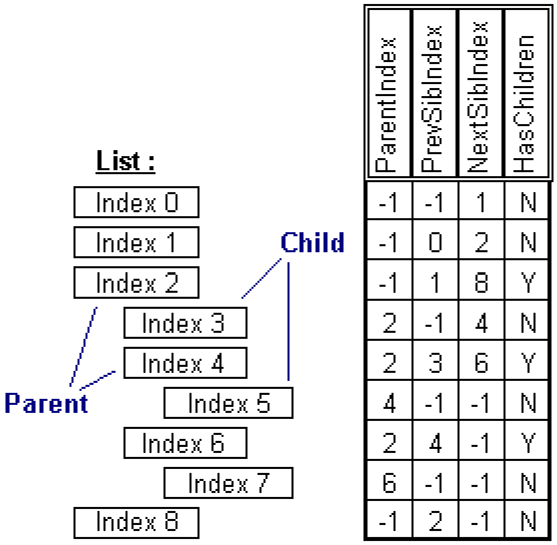
Property Window : No
Des Time : none
Run Time : Read-Write

The ListType Property

See also :

Parsing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse

Property Type :

Integer (Enum), [0-2]

Purpose :

determine current working list (tokens, keywords, symbols)

Description :

This property is designed to allow you access to the Symbols, Keywords, and Parsed String Lists. By changing the ListType Property, you can access all the lists using the same List Interface (ListCount, List(), etc.)
The ListType Property determines the current working list for the LParse. The three different lists available are as follows :

0 - the Parsed String/Token List
1 - the Keywords List
2 - the Symbols List

Example :

```
See example in ParseString
```

Property Access :

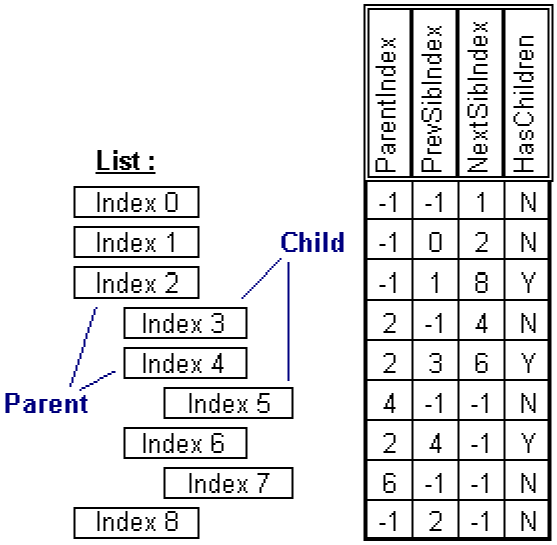
Property Window : Yes
Des Time : Read-Write
Run Time : Read-Write

The MaxTokenLen Property

See also :

Parsing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse

Property Type :

Integer

Purpose :

determines max token length

Description :

The MaxTokenLen Property specifies the maximum number of characters a token is allowed to be. The default value for this property is 256 characters. If Token is found longer than this number of characters, the Token will be truncated at that point and will result in two or moreTokens.
This property is mainly used to keep the control from allocating an inordinate amount of memory for most pasring operations. You should set this property's value to a realistic expectation of the longest meaningful token to be found. The default should suffice in most cases.

Example :

```
- Setting Values with Code :
' Set MaxTokenLen to Handle Larger Tokens
LParse.MaxTokenLen = 512

- Getting Values with Code :
Dim I As Integer
I = LParse.MaxTokenLen
```

Property Access :

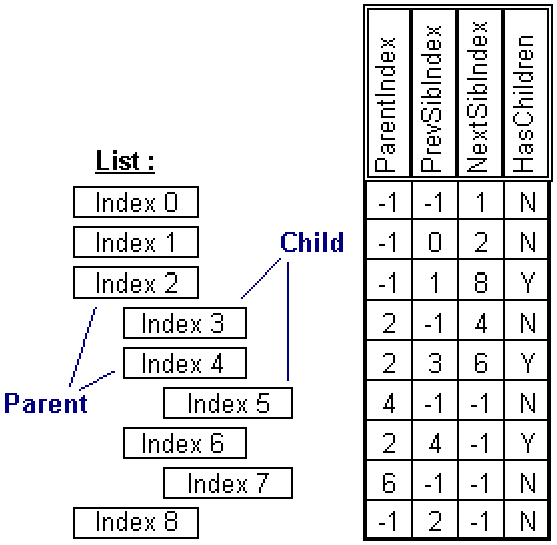
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The MinScroll Property

See also :

List Display Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas

Property Type :

Integer

Purpose :

minimum amout to scroll horizontally and vertically

Description :

When scrolling the control, this property specifies number of pixels the control will scroll.   This is used when scrolling with the scrollbars' arrow buttons or using the border's scrolling featur.

Property Access :

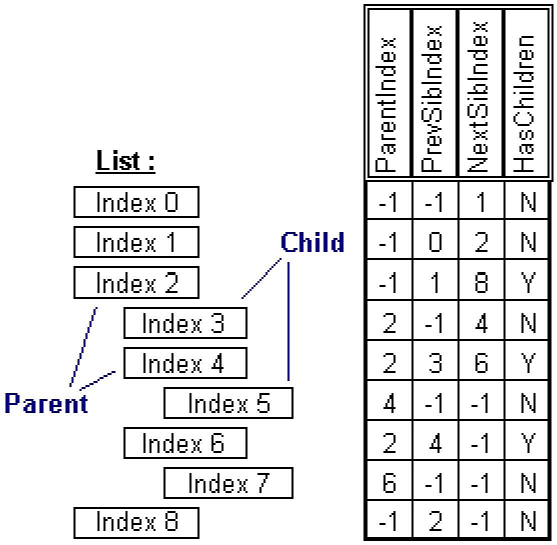
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The MoveCurr Property

See also :

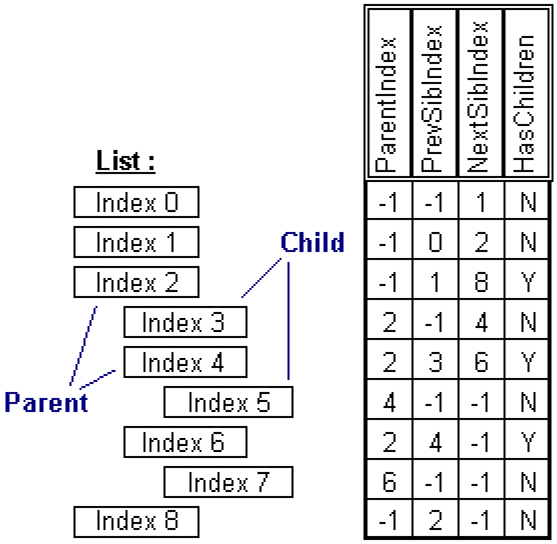
Tree Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas

Property Type :

Integer (Enum), [0-6]

Purpose :

change the current node

Description :

By changing the Current Node, the control will automatically be able to access the new Current Child List with the List and Item Properties. This property is also useful when traversing the tree of nodes, especially when using recursion.
The MoveCurr Property allows you to programmatically set the Current Node. The Current Node (starts out as the Root Node), can be selected by assigning a value, 0 thru 6, to the MoveCurr Property.

(0) No Move
(1) Move to the First Node in the Current Sibling List
(2) Move to the Last Node in the Current Sibling List

(3) Move to the Previous Node in the Current Sibling List
(4) Move to the Next Node in the Current Sibling List
(5) Move to the Parent Node of the Current Node
(6) Move to the Root Node (always valid)

When the control is first loaded and created, the Current Node is the Root Node (which always exists and can never be deleted). This means that all List Manipulation through the List and Item Properties will be performed on the list of children of the Root. By changing the Current Node, you may access children of those children, and so on.

If you try to use MoveCurr to move to a nonexistant node, the BadMove Property will be set to True (it is reset to False on all successful moves)..

Example :

```
- Setting Values with Code :
Dim Count As Integer
' Reset MTree Error Status
MTree.BadMove = False
' Set Current Node to First Node in Sibling List
MTree.MoveCurr = 1
' Loop through All Nodes in Sibling List
' and list the CurrString for each in a Listbox
List.Clear
While Not MTree.BadMove
    ' Add Data from Current Node to a Listbox
    List.AddItem MTree.CurrString
    ' Move to Next Node in Sibling List
    MTree.MoveCurr = 4
Wend
```

```
- Getting Values with Code :
Dim I As Integer
' Get the Last Move Operation that was Performed
I = MTree.MoveCurr
```

Property Access :

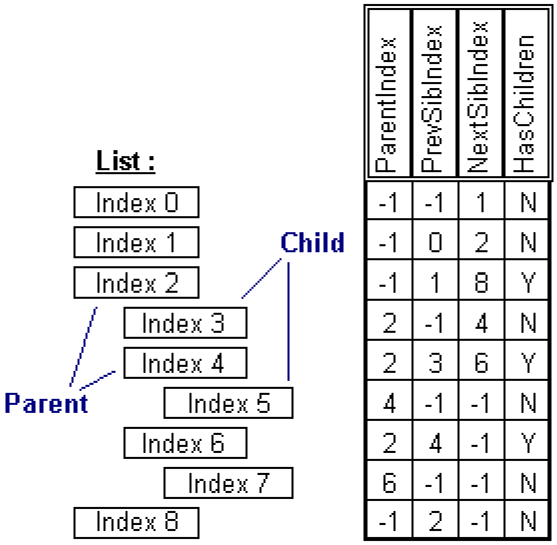
Property Window : No
Des Time : none
Run Time : Read-Write

The MoveToChild Property

See also :

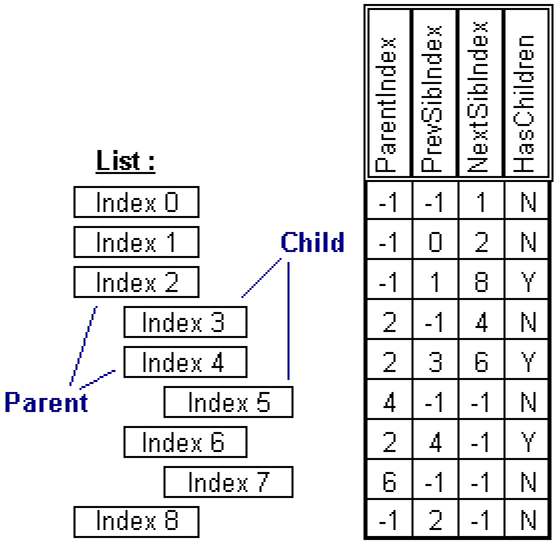
Tree Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas

Property Type :

Long

Purpose :

move the current node to a child node

Description :

By changing the Current Node, the control will automatically be able to access the new Current Child List using the List and Item Properties.   This property is also useful when traversing the tree of nodes, and especially when using recursion.
The MoveToChild Property allows you to change the Current Node in a Tree, to one of the nodes in the Current List of Children.
This is done by assigning the index of the child to the MoveToChild Property.

Example :

```
- Setting Values with Code :
' Move to the First Child in the ChildList
Tree.MoveToChild = 0
```

```
- Getting Values with Code :
Dim I As Integer
' Get the Last Value Used with MoveToChild
I = Tree.MoveToChild
```

Property Access :

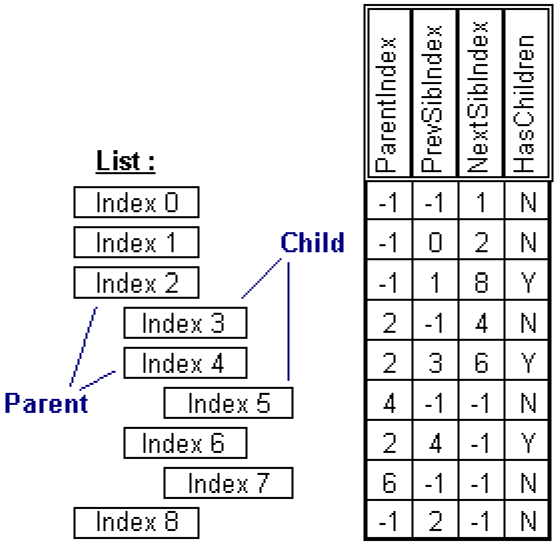
Property Window : No
Des Time : none
Run Time : Read-Write

The NewIndex Property

See also :

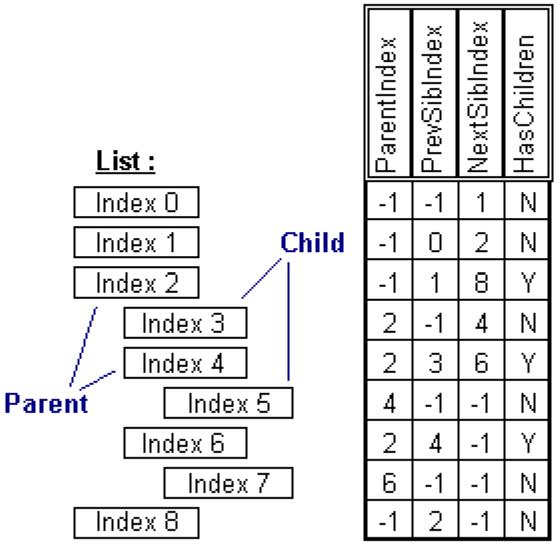
List Data Properties

Applies To :



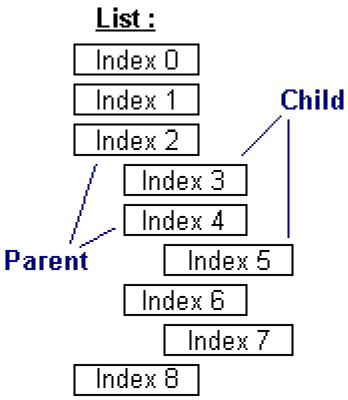
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



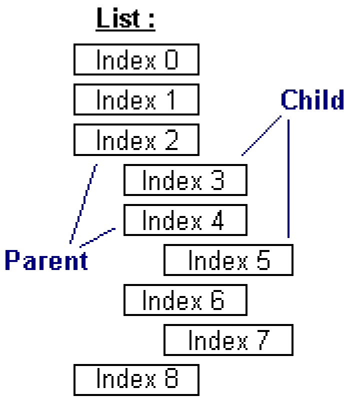
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList



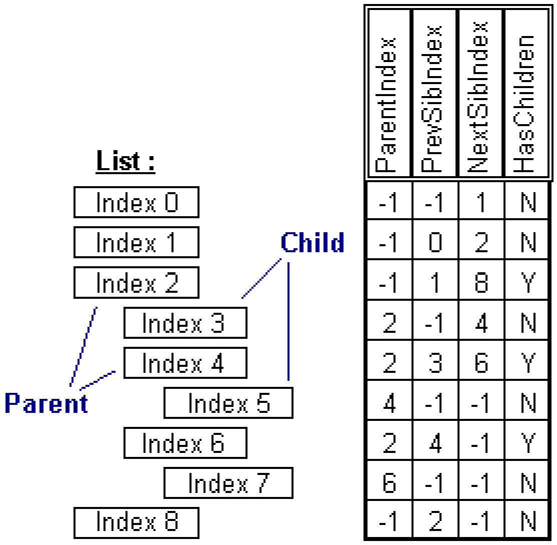
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



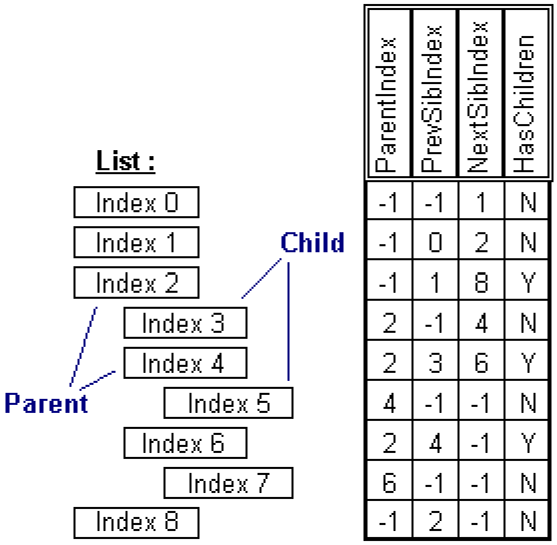
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse



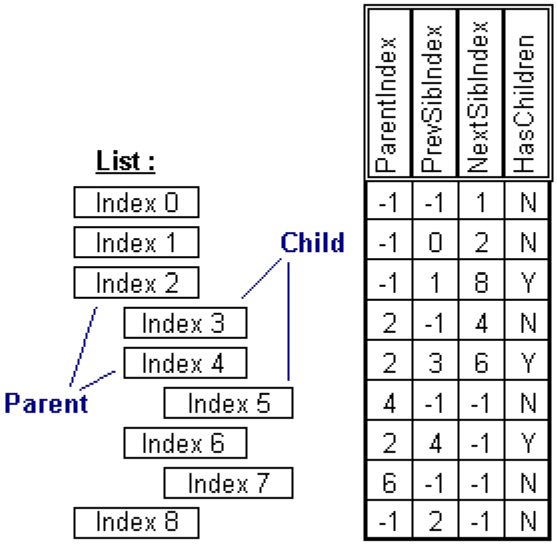
ParentIndex	PrevSiblndex	NextSiblndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



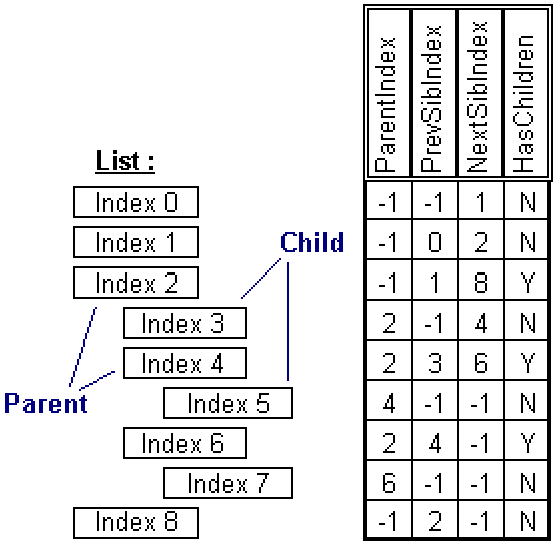
ParentIndex	PrevSiblndex	NextSiblndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Long

Purpose :

Index of Most Recently Added Item in List

Description :

This property is handy when setting the ItemData and/or ItemString properties of Items that have just been added to a List.

Example :

```
- Setting Values with Code :
This Property is Read-Only at Run-Time

- Getting Values with Code :
Dim I as Integer
I = List.NewIndex
' The Following Lines Add an Item and then
' sets the ItemString of that Item
List.AddItem "New Item in the List"
List.ItemString( List.NewIndex ) = "New Extra String"
```

Property Access :

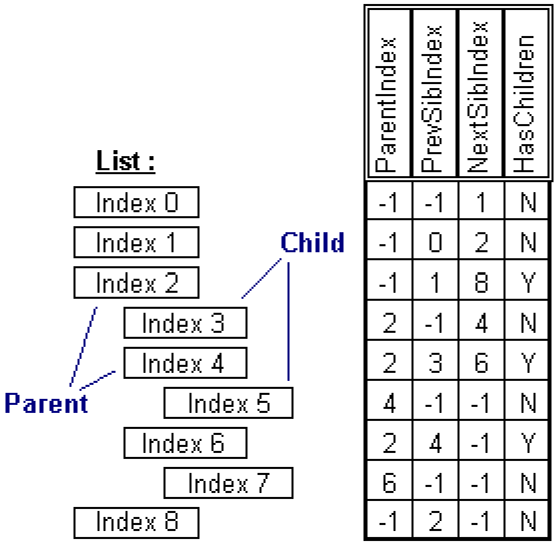
Property Window : No
Des Time : Read-Write
Run Time : Read-Only

The NextSibIndex Property

See also :

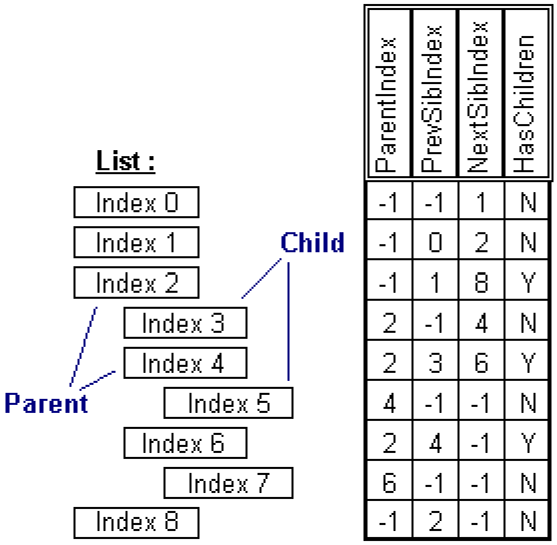
Tree Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Long, Property Array

Purpose :

returns the Index of the next-most Item at the same Indent Level

Description :

Through the use of Indentation within a List, a hierarchical parent-child relationship is established.   The NextSibIndex Property returns the Index of the next-most Item, if any, that has the same Parent as the specified Item.   If no next sibling exists for that Item, a -1 is returned.

This property is used when maintaining a tree structure within a List.   By finding the PrevSibIndex and Next SibIndex of an Item, you are able to traverse only the List of Items which occur at the same indent level and have the same Parent Item.

Example :

```
- Getting Values with Code :
Dim I As Integer
Dim S As String
' Start with Item #20, for example
I = 20
' Find First Sibling
While( List.PrevSibIndex( I ) <> -1 )
    I = List.PrevSibIndex( I )
Wend
' Loop through all of the Siblings
While( I <> -1)
    ' Build a String
    S = (S + List.List( I ) + ", ")
    ' Get Next Sibling
    I = List.NextSibIndex( I )
Wend
```

Property Access :

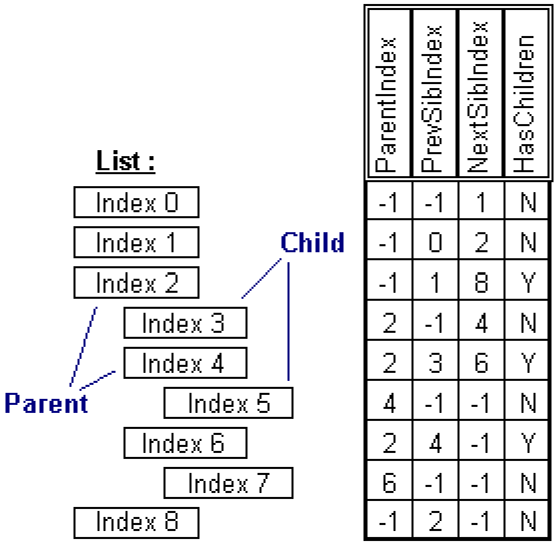
Property Window : No
Des Time : none
Run Time : Read-Write

The ParentIndex Property

See also :

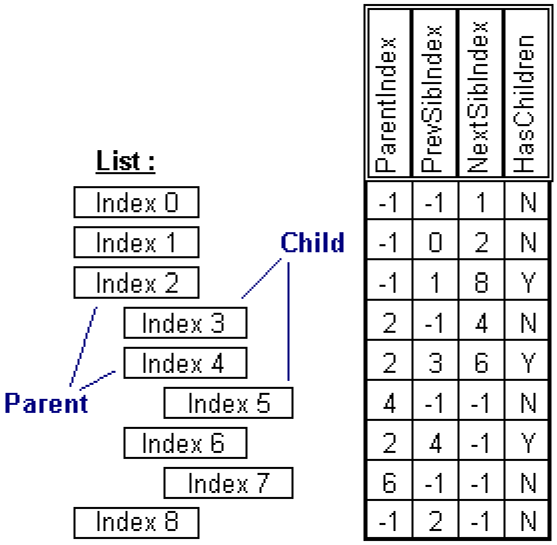
Tree Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Long, Property Array

Purpose :

returns the Index of the Parent of a List Item

Description :

Through the use of Indentation within a List, a hierarchical parent-child relationship is established. The ParentIndex Property returns the Index of the Parent Item, if any, of the specified Item. If no Parent Exists for that Item, a -1 is returned.

This property is used when maintaining a tree structure within a List. By finding the ParentIndex of an Item may be able to you what directory a file is in, or what department a particular employee is in.

Example :

```
- Getting Values with Code :
Dim I As Integer
Dim S As String
' Find the Employee "John Smith"
List.SearchString = "John Smith"
I = List.FoundIndex
' Get The Department Name for Employee
S = List.List( List.ParentIndex( I ) )
```

Property Access :

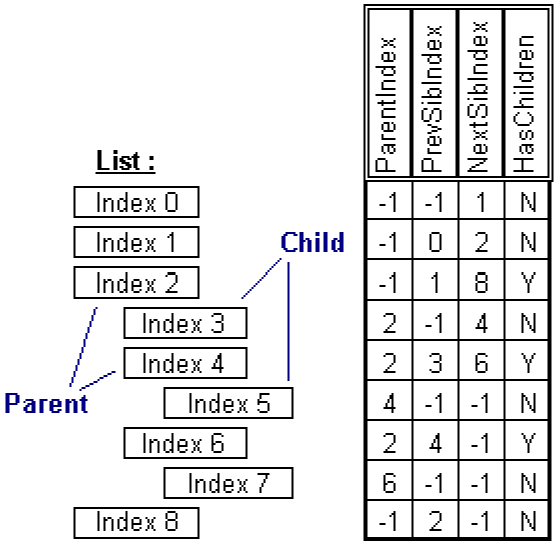
Property Window : No
Des Time : none
Run Time : Read-Write

The ParseString Property

See also :

Parsing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse

Property Type :

Boolean

Purpose :

string to be parsed

Description :

Using the ParseString Property and the Parsed String List allow you to easily identify and manipulate portions of strings. By cutting a string up into tokens, you can search for an individual token, modify it, and retrieve the modification within the rebuilt ParseString.
The ParseString Property is, after setting up the LParse, one of the main points of interaction between your program and LParse.
The setup of LParse is accomplished by adding WhiteSpace Characters, Symbols, and Keywords, and then setting the various "b" Flags. In order to parse tokens out of a string, assign that string to the ParseString Property.   LParse will then begin searching and identifying tokens (text delimited by whitespace and symbols), and adding them to the Parsed String List (ListType = 0).
After the string has been parsed, LParse will recreate the ParseString by recombining the items in the Parsed String List. This occurs every time you ask for the value of the ParseString Property. Therefore, any changes made to the Items in the Parsed String List will be reflected the next time you get the value of the ParseString.

Example :

```
Setting Values with Code :
' Setup to Parse FileNames :
' No Keywords
LParse.ListType = 1
LParse.Clear
' Add Some Symbols
LParse.ListType = 2
LParse.Clear
LParse.AddItem ":\"
LParse.AddItem "\"
LParse.AddItem "."
' Reset to Parsed String List
LParse.ListType = 0
LParse.Clear
' No Whitespace Allowed
LParse.WSpaceChars = ""
' Example FileName :
LParse.ParseString = "C:\Dos\Command.COM"
' The Parsed String List Should Look Like :
'  C                  : Identifier
'  :\                 : Symbol #0
'  Dos                : Identifier
'  \                  : Symbol #1
'  Command            : Identifier
'  .                  : Symbol #2
'  COM                : Identifier

- Getting Values with Code :
Dim S As String
S = LParse.ParseString
```

Property Access :

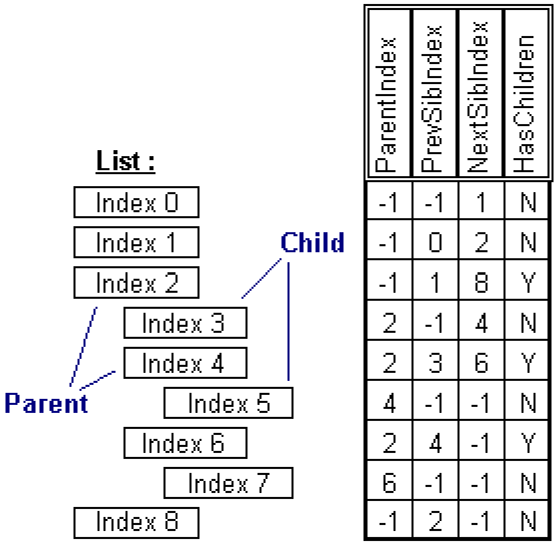
Property Window : No
Des Time : Read-Write
Run Time : Read-Write

The PathChar Property

See also :

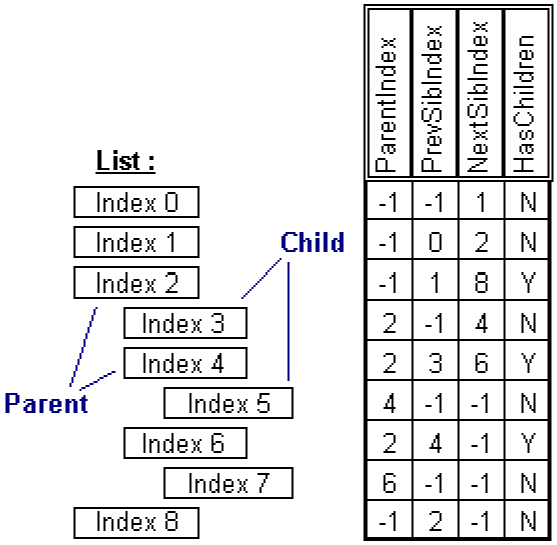
Tree Path Properties

Applies To :



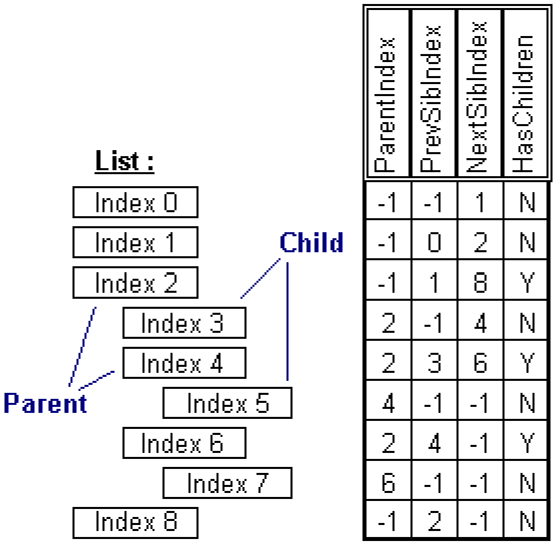
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :
String, single character

Purpose :
the char to be used for getting and setting paths

Description :
The PathChar Property allows you to specify the delimiting character used in building the RelPath Property. The default value for this property is the back-slash '\'. If this property is set to nothing (i.e. empty string ""), the value will revert to the default.

Note: This property is useful to customize the generation of a path-like string for a given item in the list.

Example :
```
- Setting Values with Code :
Dim S As String
' Set the PathChar to a Comma
List.PathChar = ","
' Add Some Items to the List
List.AddItem "ItemA"
List.AddItem "ItemB"
List.ItemIndent(List.NewIndex) = 1
List.AddItem "ItemC"
List.ItemIndent(List.NewIndex) = 2
List.AddItem "ItemD"
List.ItemIndent(List.NewIndex) = 2
List.AddItem "ItemE"
List.ItemIndent(List.NewIndex) = 1
' The Resulting List would Look Like :
'   ItemA
'      ItemB
'         ItemC
'         ItemD
'      ItemE
' Get Path of 'ItemD'
S = List.RelPath(3)
' Path would Look Like :
'   "ItemA,ItemB,ItemD"

- Getting Values with Code :
Dim S As String
S = List.PathChar
```

Property Access :

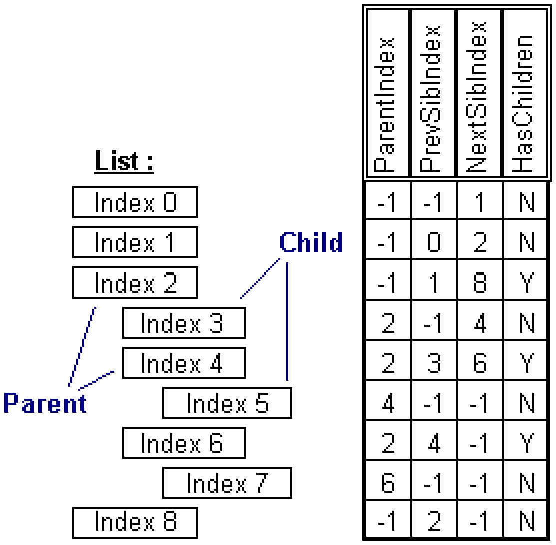
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The PopUp Property

See also :

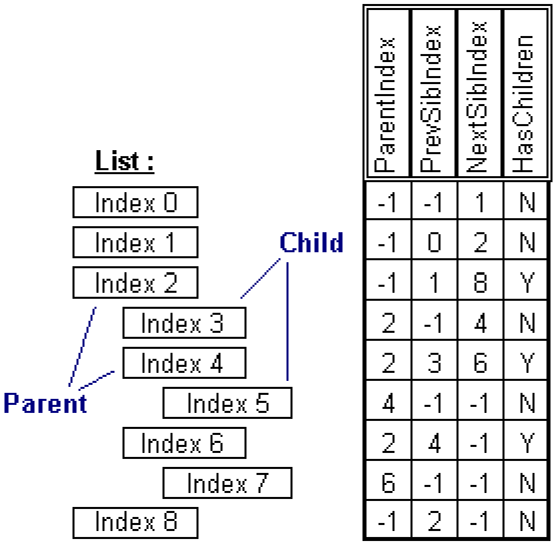
Control Properties

Applies To :



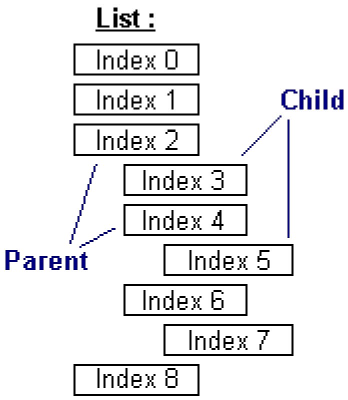
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



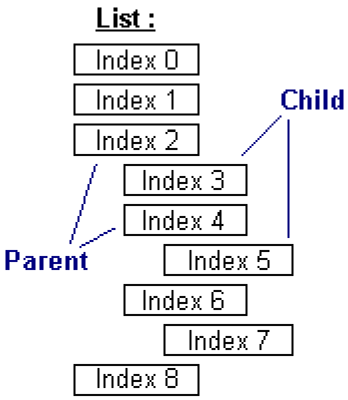
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



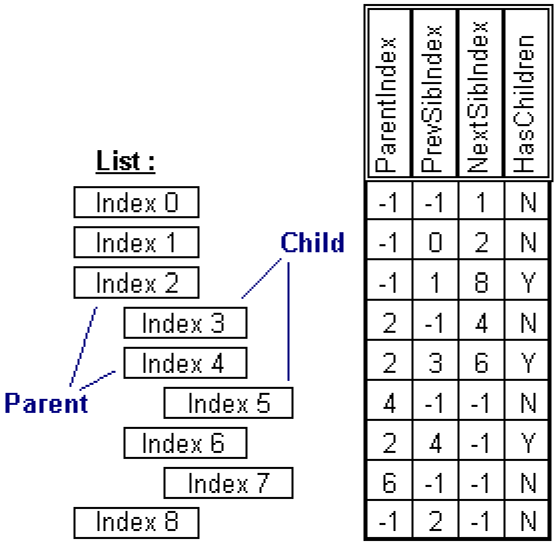
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Boolean

Purpose :

set to use control as a popup window outside of form

Description :

This property lets you create a popup window that can appear outside of its own form. There are some differences when operating in popup mode that are worthy of note. First, the Top and Left properties will now use coordinates relative to the screen instead of the parent form. Secondly, the control is forced to be always on top (this is tough when trying to display message boxes that appear behind the control). Lastly, the control does not draw focus rectangles and any multiple selection is disabled. Even with some of the disadvantages, using this property in combination with the TrueShadow is a totally cool way to display a quick selection of items.

Example :

```
lblFile.ControlFlags = "+BRD_PUSHED"
lblFile.Refresh
lstFile.PopUp = True
```

Property Access :

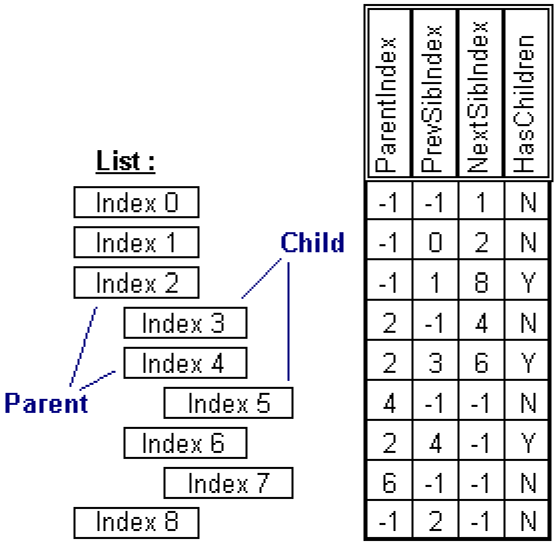
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The PrevSibIndex Property

See also :

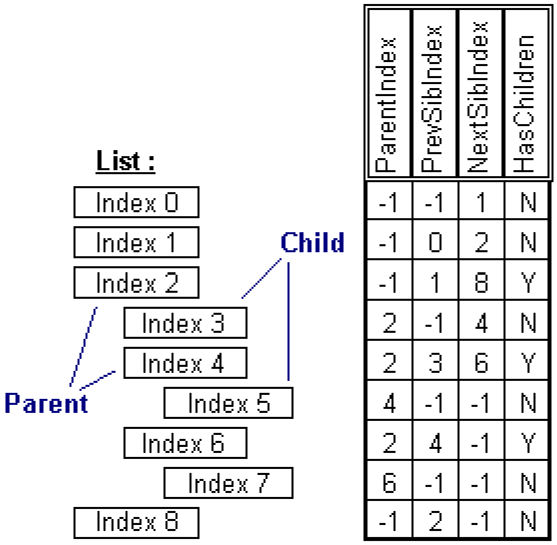
Tree Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Long, Property Array

Purpose :

returns the Index of the most previous Item at the same Indent Level

Description :

Through the use of Indentation within a List, a hierarchical parent-child relationship is established. The PrevSibIndex Property returns the Index of the most previous Item, if any, that has the same Parent as the specified Item. If no previos sibling exists for that Item, a -1 is returned.

This property is used when maintaining a tree structure within a List. By finding the PrevSibIndex and Next SibIndex of an Item, you are able to traverse only the List of Items which occur at the same indent level and have the same Parent Item.

Example :

```
- Getting Values with Code :
Dim I As Integer
Dim S As String
' Start with Item #20, for example
I = 20
' Find First Sibling
While( List.PrevSibIndex( I ) <> -1 )
    I = List.PrevSibIndex( I )
Wend
' Loop through all of the Siblings
While( I <> -1)
    ' Build a String
    S = (S + List.List( I ) + ", ")
    ' Get Next Sibling
    I = List.NextSibIndex( I )
Wend
```

Property Access :

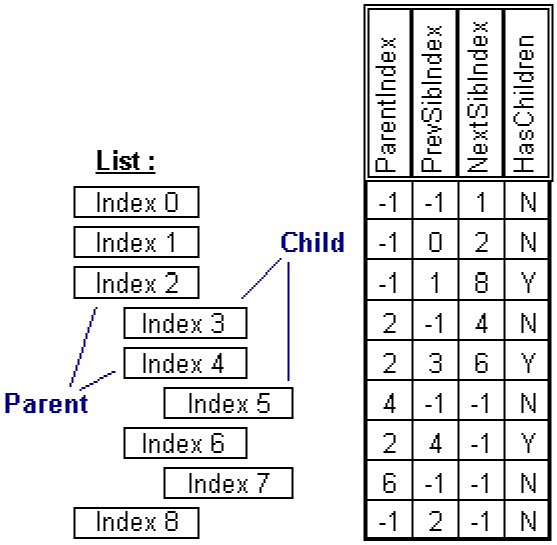
Property Window : No
Des Time : none
Run Time : Read-Write

The RawText Property

See also :

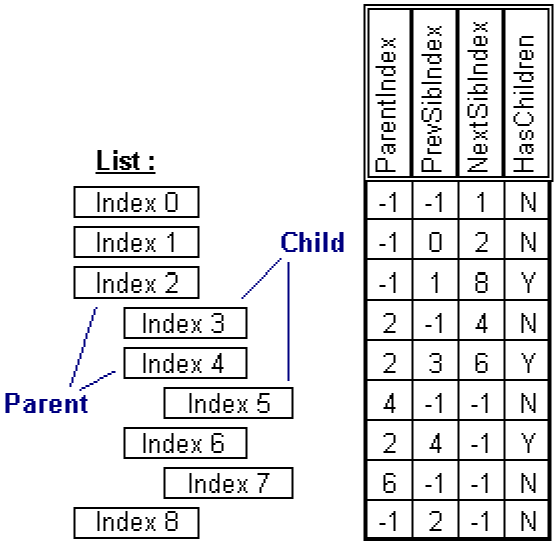
Item Data Properties

Applies To :



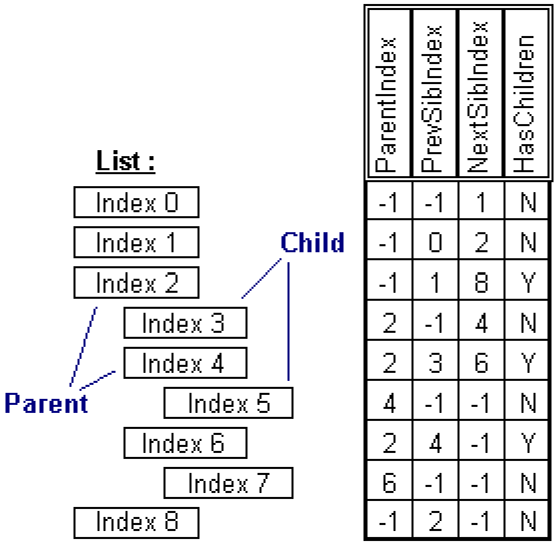
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

String, Property Array

Purpose :

Text Sans Formatting

Description :

The RawText Property is a Property Array that returns, for each Item in the List, the text of that Item minus any formatting codes. That is, if the text of an Item contains any embedded formatting codes, the RawText Property will give you the text without those codes.

This property is useful when you need to get the text of an Item and do something with it.   If you were displaying a directory tree, with the directory names in blue and the filenames in bolded red, the RawText Property could be used to get just the filename or directory name.

Example :

```
- Getting Values with Code :
Dim S As String
' Get the FileName of the Current Item
S = List.RawText( List.ListIndex )
' Call a Sub to Open the File
Call OpenThisFileName( S )
```

Property Access :

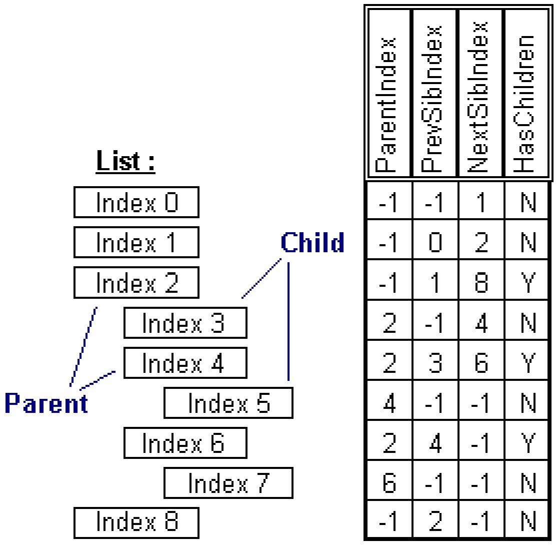
Property Window : No
Des Time : Read-Write
Run Time : Read-Only

The RelPath Property

See also :

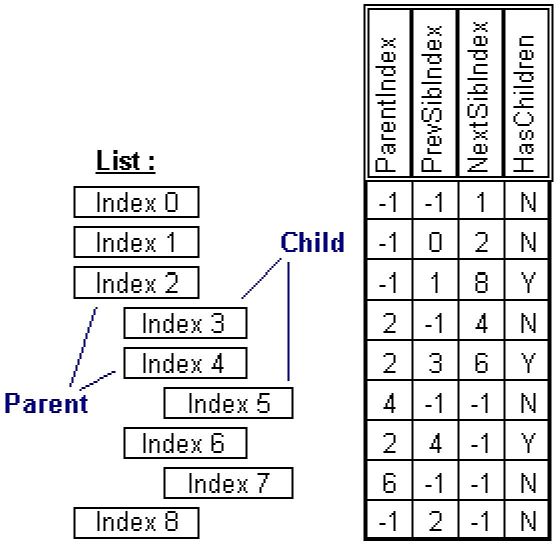
Tree Path Properties
AbsPath

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

String, Property Array

Purpose :

an item's path constructed with an item's text

Description :

The RelPath Property returns a path-like string for a given Item in the List. To build this string, the List starts with the specified Item, and works its way up through the Parent Items, adding the RawText for each Item as it goes.

When the RelPath string is being built, the control uses the PathChar Property to seperate the text for each Item in the path. The text for each Item used in the RelPath string is the RawText Property for that Item.

This property is useful when you need to obtain a path-like string for a given Item. If you were using the List to display disk directories and files, the RelPath Property would give you a DOS type path string for an Item (given that PathChar was set to '\').

Example :

```
- Getting Values with Code :
Dim S As String
' Set the PathChar to a SemiColon
List.PathChar = ";"
' Add Some Items to the List
List.AddItem "ItemA"
List.AddItem "ItemB"
List.ItemIndent(List.NewIndex) = 1
List.AddItem "ItemC"
List.ItemIndent(List.NewIndex) = 2
List.AddItem "ItemD"
List.ItemIndent(List.NewIndex) = 2
List.AddItem "ItemE"
List.ItemIndent(List.NewIndex) = 1
' The Resulting List would Look Like :
'   ItemA
'      ItemB
'         ItemC
'         ItemD
'      ItemE
' Get RelPath of 'ItemD'
S = List.RelPath(3)
' Path would Look Like :
'   "ItemA;ItemB;ItemD"
' Get AbsPath of 'ItemD'
S = List.AbsPath(3)
' Path would Look Like :
'   "0.1.3"
```

Property Access :

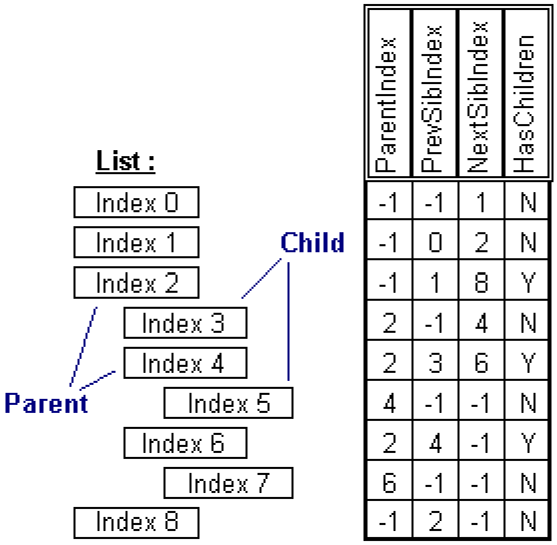
Property Window : No
Des Time : none
Run Time : Read-Only

The ScrollWidth Property

See also :

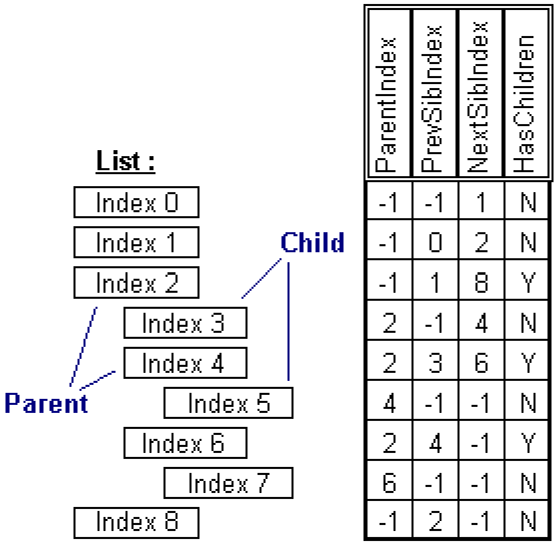
List Display Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList

Property Type :

Integer

Purpose :

sets the scrollable width for a control

Description :

The ScrollWidth Property allows you to display a Horizontal ScrollBar so the List may be Scrolled Left and Right. This Property takes an Integer value representing the width you wish to display in numbers of characters.

This Property is useful when you know that you will be displaying rather long Items in your List, such as PathNames or whole Database Records. If you were displaying records from a database wherein each record was 378 characters long, you would want to set the ScrollWidth equal to 378. The List will automatically calculate the display width to use considering the current Font and

FontSize.   Also, a Horizontal ScrollBar will only appear when the display width exceeds the visible width of the control..

Example :

```
- Setting Values with Code :
Dim I As Integer
Dim CurrWidth As Integer
List.ScrollWidth = 0 ' Disable Horizontal Scroll
' Add a Bunch of Items
For I = 1 To 1000
    ' Add an Item
    List.AddItem ('Some String that may be Really Long #" + Str$(I))
    ' Get Length of Item in Characters
    CurrWidth = Len(List.List(List.NewItem)
    ' Change ScrollWidth if Currently Not Large Enough to Hold New String
    If (CurrWidth > List.ScrollWidth) Then
        List.ScrollWidth = CurrWidth ' Set Width to Hold Longest String
    End If
Next I
```

Property Access :

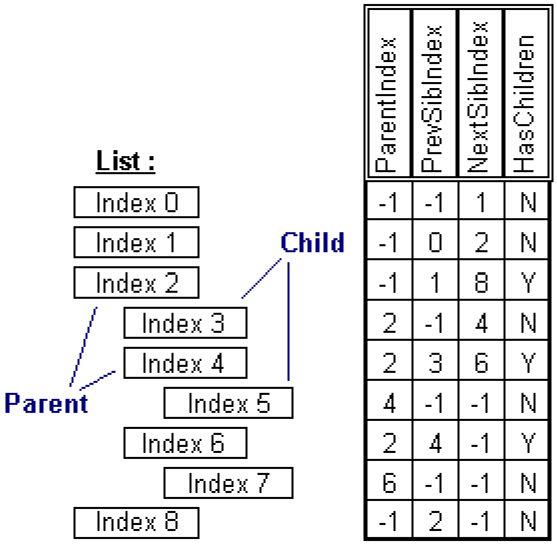
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The SearchString Property

See also :

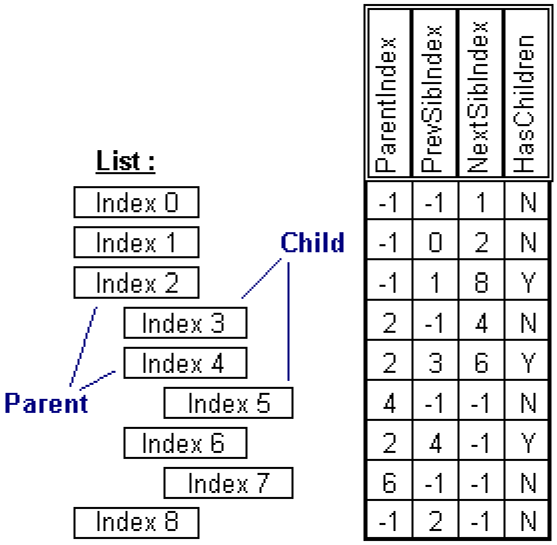
List Searching Properties

Applies To :



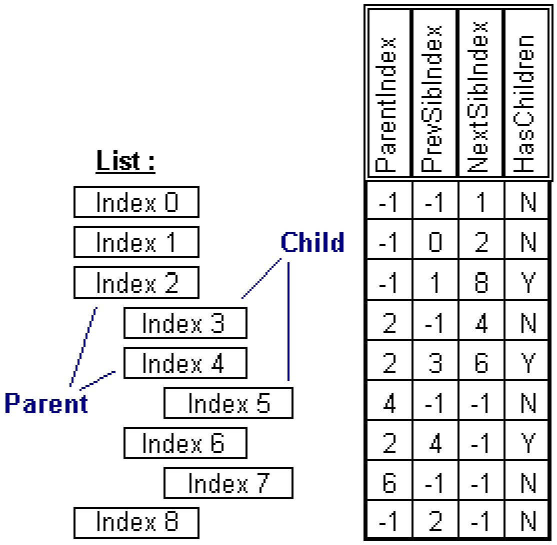
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



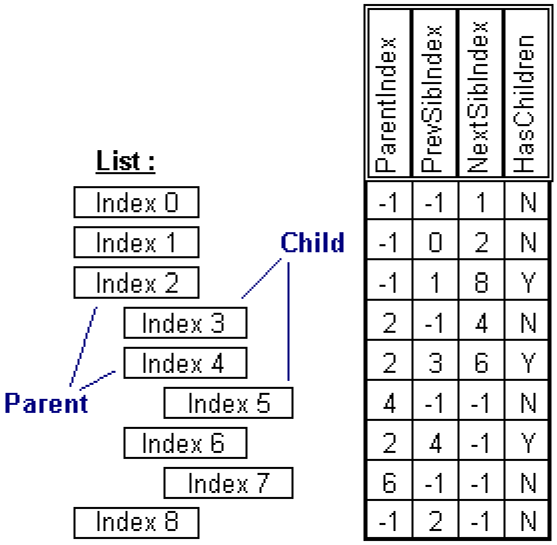
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList



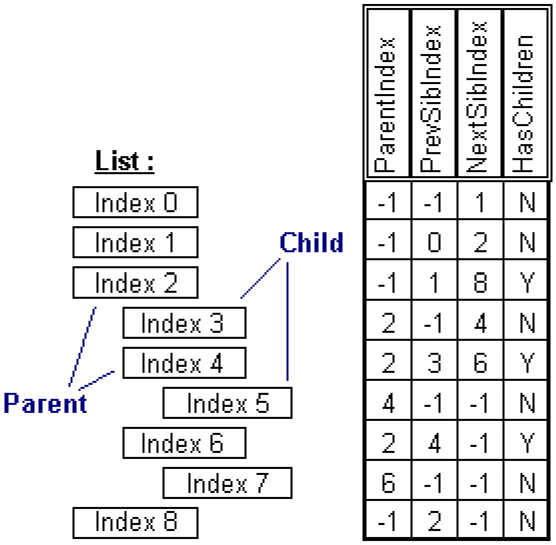
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



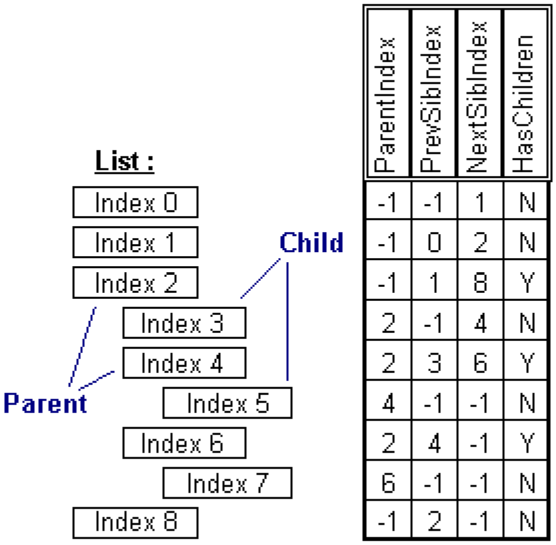
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse



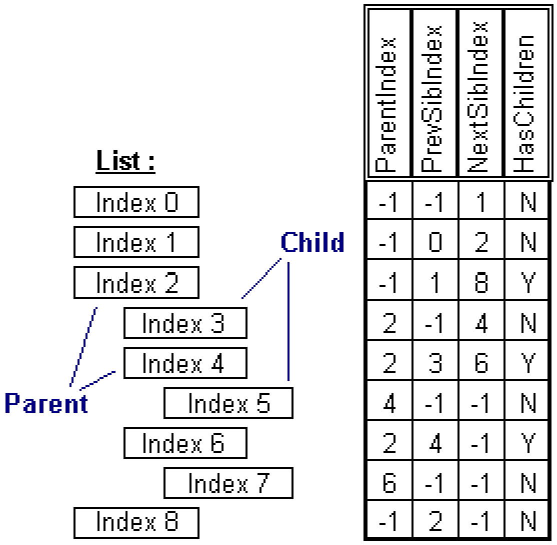
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



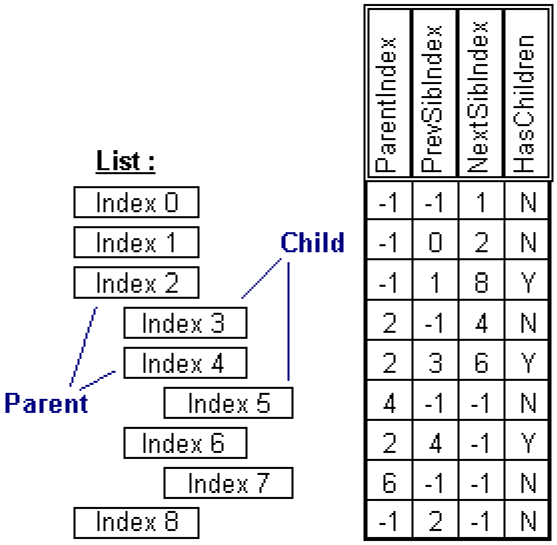
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

String

Purpose :

This property is used to locate a particular item-string in the List.

Description :

The SearchString Property is used with the StartSearch and FoundIndex Properties. When using the SearchString Property, the List will begin searching for the specified string at Index specified by the StartSearch property. If the string is found in the List, FoundIndex will contain the index at which the string was found. If the string was not found in the List, then FoundIndex will be set to -1.

Example :

```
- Setting Values with Code :
Dim S As String
List.SearchString = S
If (List.FoundIndex <> -1) Then
    MsgBox ("Found string : " + S + " at " + Str$(EnhList.FoundIndex))
Else
    MsgBox ("Could not find the string : " + S)
End If
```

```
- Getting Values with Code :
' Find All Instances of a String in the List
Dim S As String
S = "USA"
List.StartSearch = 0
List.SearchString = S
While (List.FoundIndex <> -1)
    List.StartSearch = List.FoundIndex + 1
    List.SearchString = S
Wend
```

Property Access :

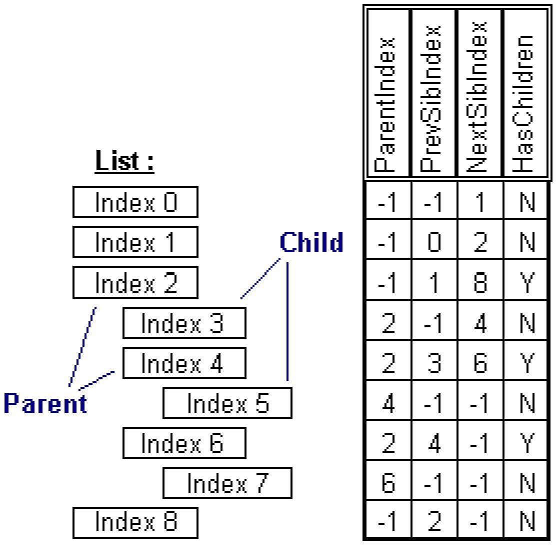
Property Window : No
Des Time : none
Run Time : Read-Write

The SearchType Property

See also :

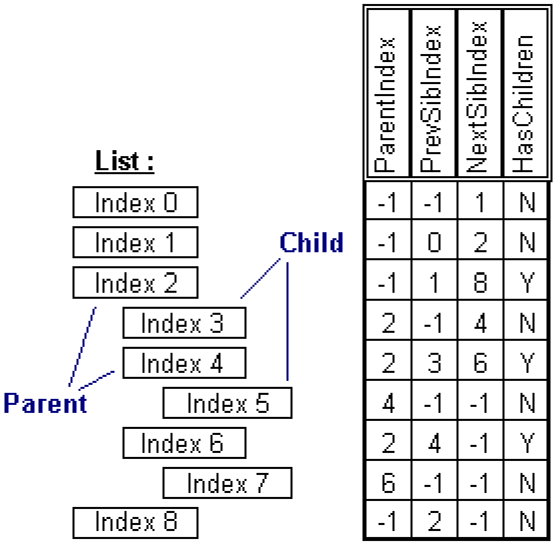
List Searching Properties

Applies To :



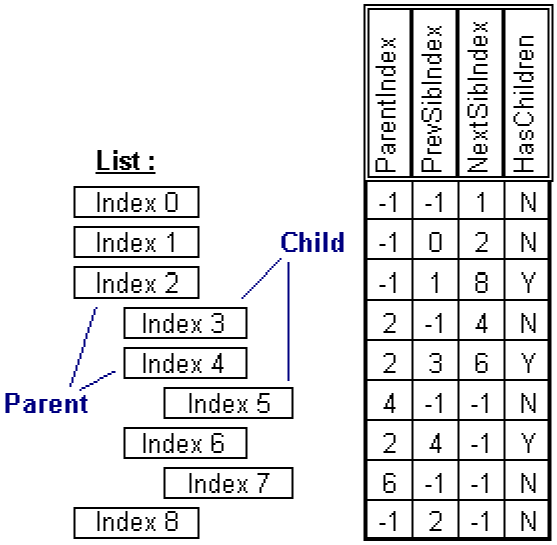
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



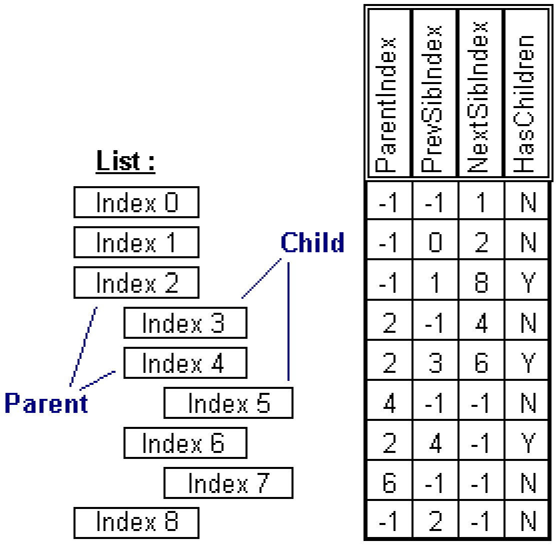
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList



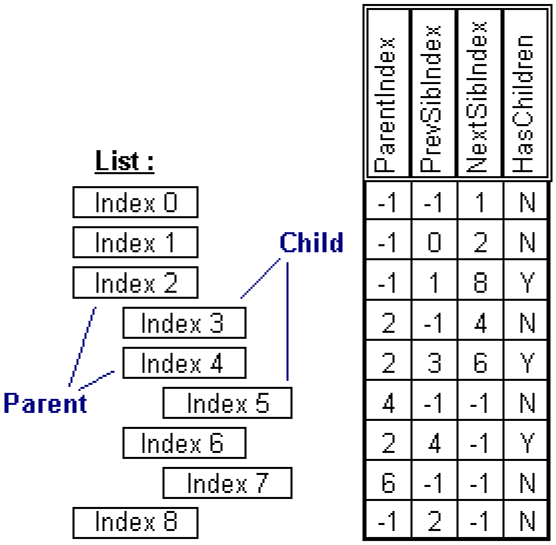
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



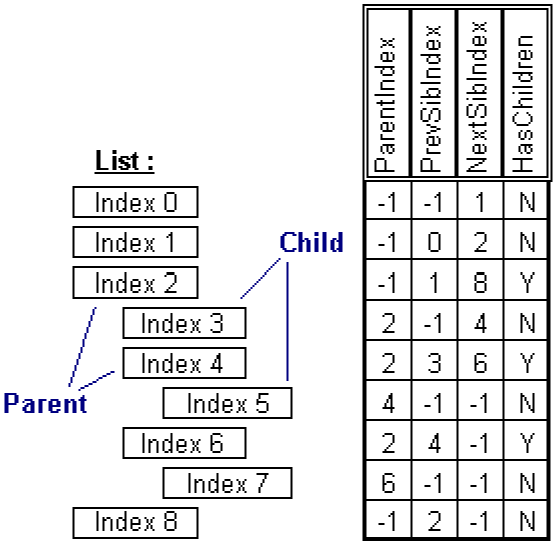
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse



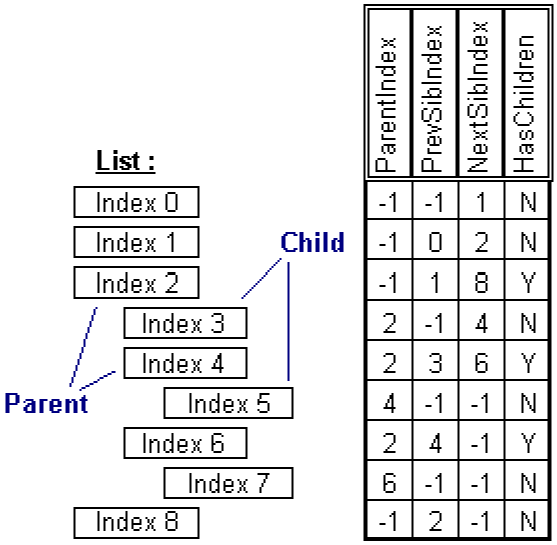
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



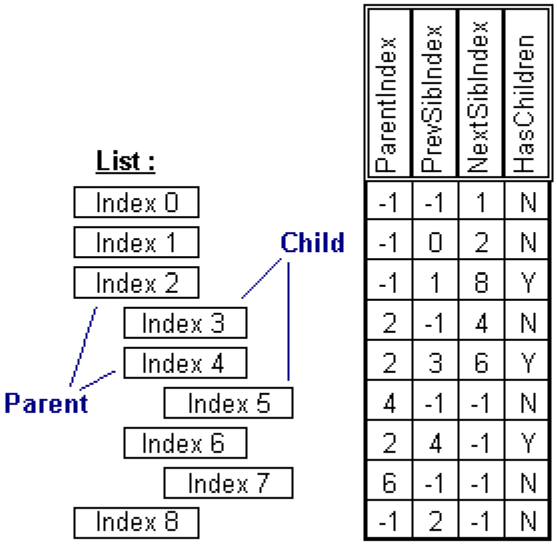
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Integer (Enum), [0-2]

Purpose :

when using the ItemData or ItemString Properties to store additional information about an Item in your list, it is often useful to have varying ways of querying your data

Description :

The SearchType Property allows for flexibility when querying the list for information. There are three different Search Types :
0 - List : Searches the List Property of a list when searching. (Note : for the CTree, the RawText Property is searched instead)
1 - ItemData : Matches the SearchString Property (converted from a string to a number), with the ItemData Property of the list.
2 - ExtraString : Uses the ExtraString Property of the list to do searches.

Example :

```
- Setting Values with Code :
Dim I As Integer
' Search the List's ItemData Property
List.SearchType = 1
' Find the First Instance where ItemData = 5
List.StartSearch = -1
List.SearchString = "5"
' Get the Index that "5" was Found At (-1 if not found)
I = List.FoundIndex
```

```
- Getting Values with Code :
Dim I As Integer
I = List.SearchType
```

Property Access :

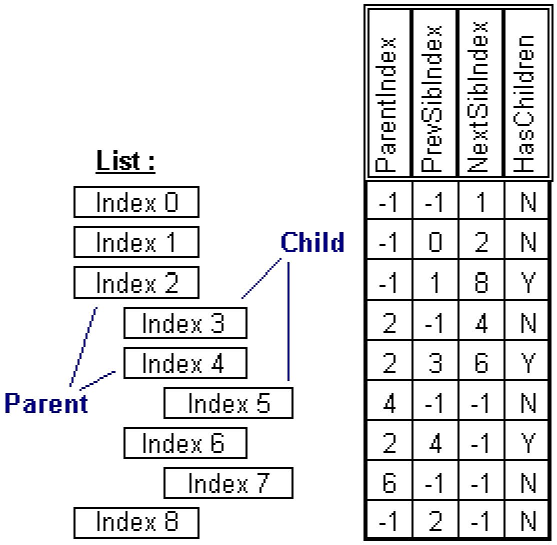
Property Window : No
Des Time : none
Run Time : Read-Write

The ShadowWidth Property

See also :

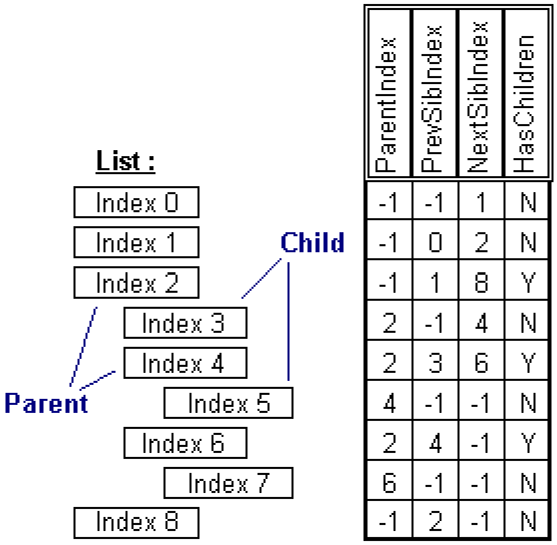
Control Border Properties

Applies To :



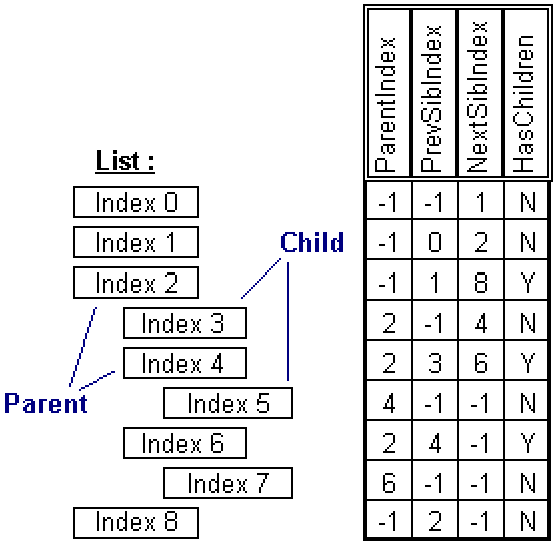
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



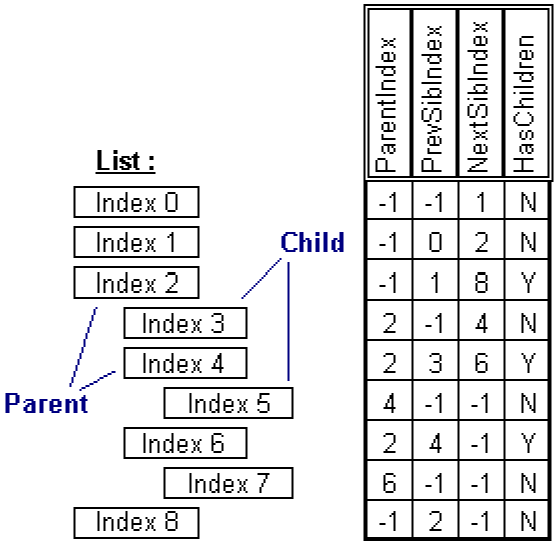
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



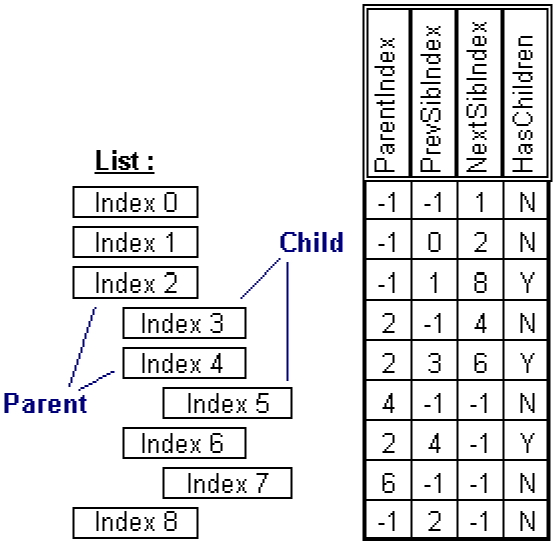
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :
Integer

Purpose :
Width of Shadow

Description :
The ShadowWidth Property allows you to specify the width of the shadow surrounding your control. This width is applied to the right and bottom of the control.

Note that the BRD_SHADOW flag must be set in the ControlFlags to show the shadow at all.

Example :
```
- Setting Values :
ctrl.ShadowWidth = 15 ' A Large Shadow
' Set Flags to Show Border, Bevelling, and Shadow
ctrl.ControlFlags = "+BRD_BORDER+BRD_BEVEL+BRD_SHADOW"
```

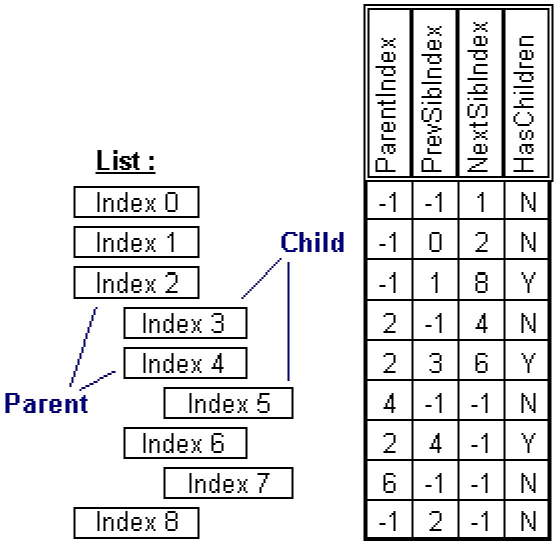
Property Access :
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The StartSearch Property

See also :

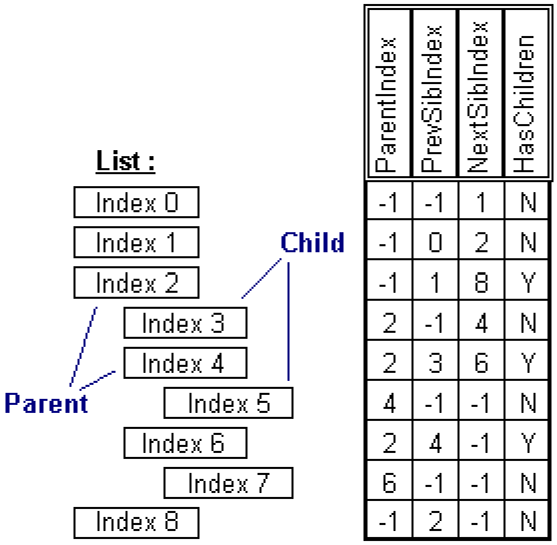
List Searching Properties

Applies To :



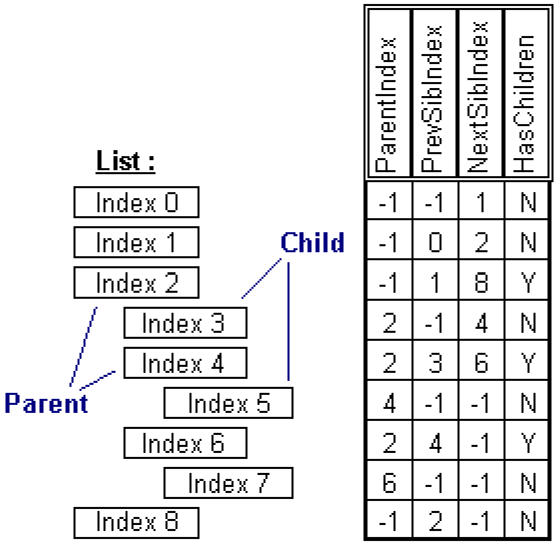
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



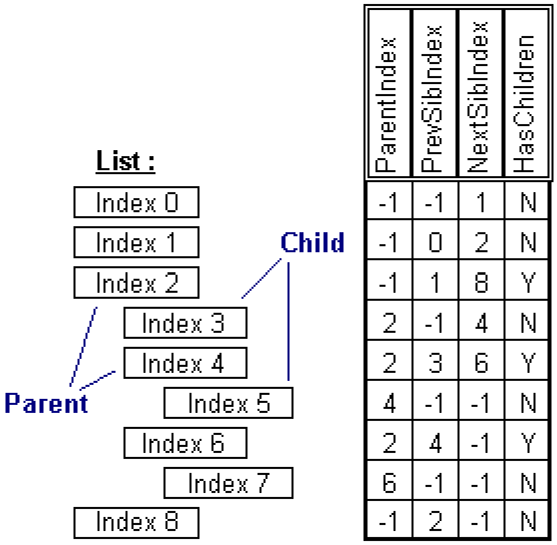
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList



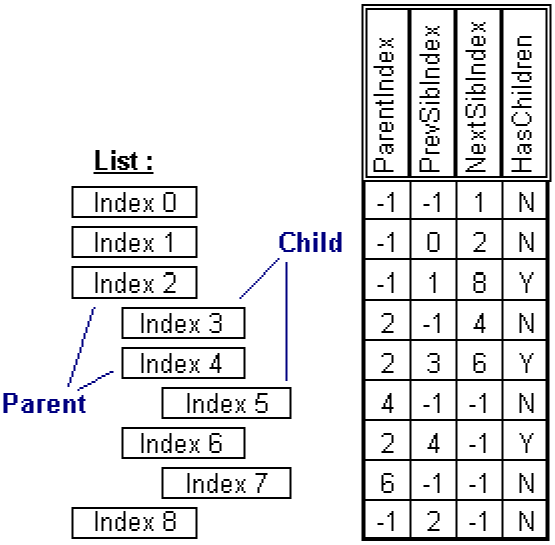
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



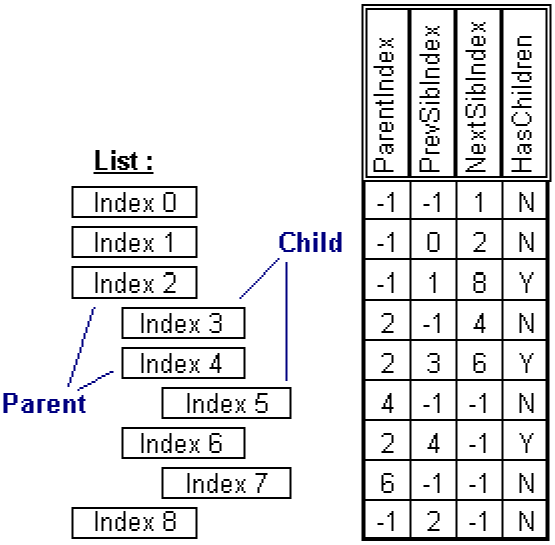
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse



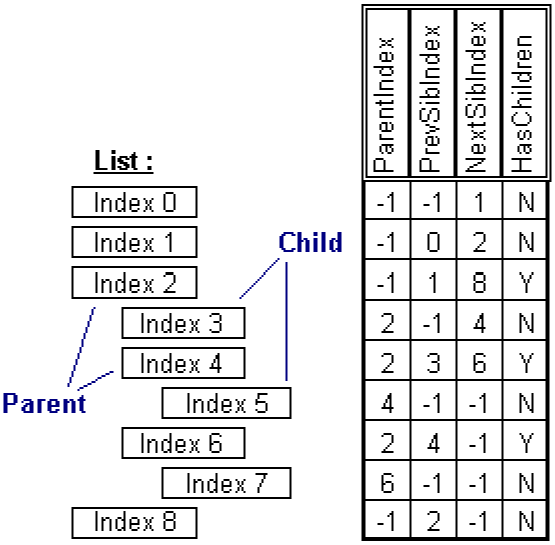
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

MTree



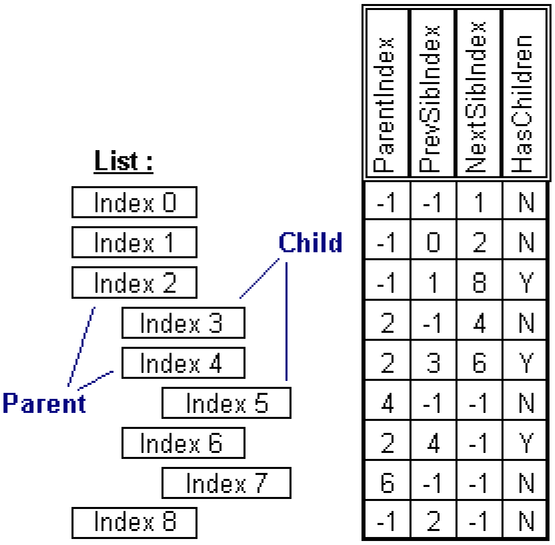
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Integer

Purpose :

useful for searching for a string that may occur more than once within the list

Description :

The StartSearch Property is used with the SearchString and FoundIndex Properties.   When using the SearchString Property, the List will begin searching for the specified string at Index specified by the StartSearch property.

Example :

```
- Setting Values with Code :
' Find All Instances of a String in the List
Dim S As String
S = "USA"
List.StartSearch = 0
List.SearchString = S
While (List.FoundIndex <> -1)
    List.StartSearch = List.FoundIndex + 1
    List.SearchString = S
Wend
```

```
- Getting Values with Code :
Dim I As Integer
I = List.StartSearch
```

Property Access :

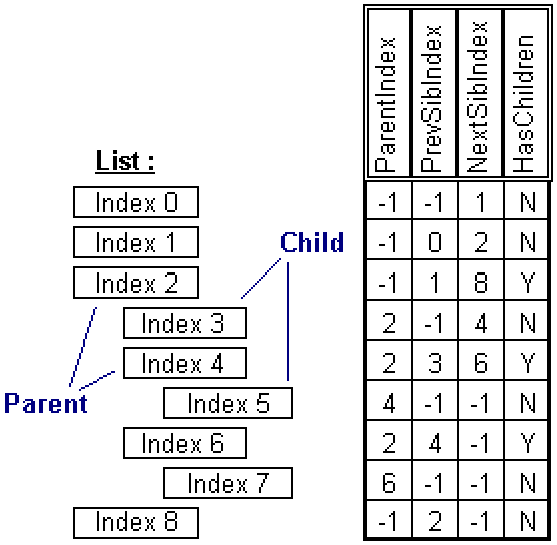
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleBackColor Property

See also :

Style Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

Long (Color), Property Array

Purpose :

determines the background color of a defined Style that may be applied to an item.

Description :

Each item in a StyleBag has a StyleName, StyleFontName, StyleFontSize, StyleBold, StyleItalic, StyleUnderline, StyleForColor, StyleBackColor, etc.

Example :

```
See example in StyleBag Property
```

Property Access :

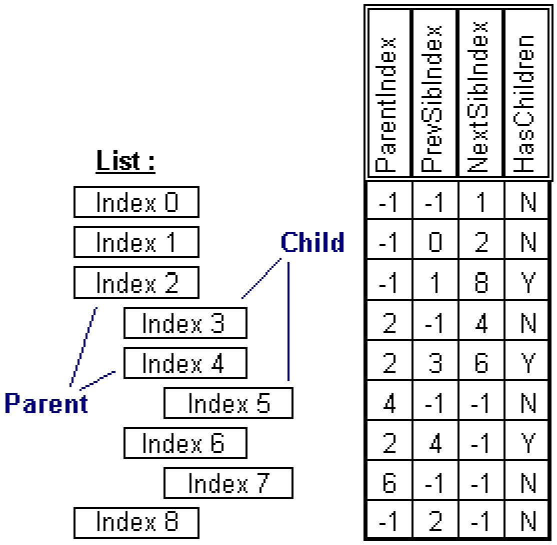
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleBag Property

See also :

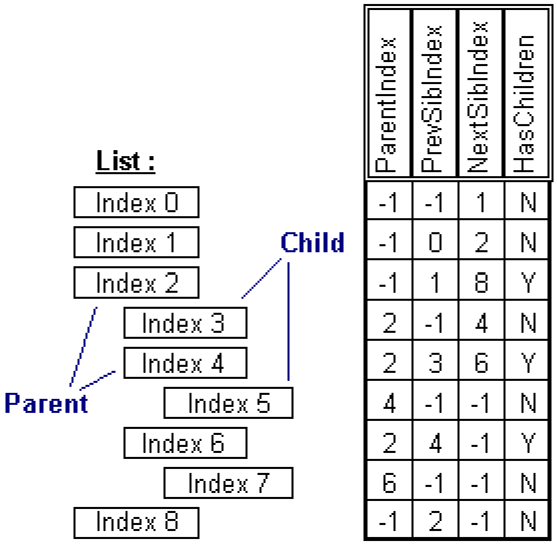
Control Properties

Applies To :



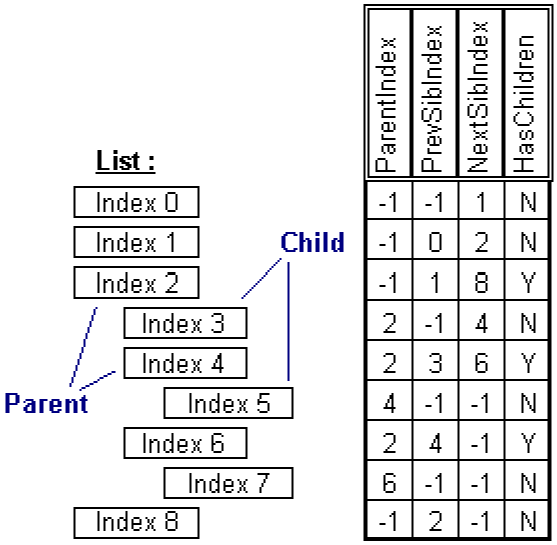
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Integer

Purpose :

hWnd of a Source StyleBag

Description :

This property identifies the StyleBag which will serve as the source for all the styles used within the control. This is much like connecting to a Data Control, except that you are identifying the StyleBag by its window handle (hWnd) rather than its name.

During design-time, you are able to set the StyleBag by entering its name in the properties window. The control will immediately identify the StyleBag's presence and store its window handle. When the control is unloaded and saved to disk at design-time, it stores the StyleBag's name and reconverts it the next time it is loaded.

Example :

```
- Setting Values with Code :
' Set up a Style to associate with the selected items in a list
' Constants
Const cBlack =&H00000000&
Const cRed = &H000000FF&
Const cYellow = &H0000FFFF&
Const cDkGrey = &H00808080&
Const cWhite = &H00FFFFFF&
Const cLtGrey = &H00C0C0C0&
Const DEFITEM_SELECTED = 2
Const DEFITEM_CURRENT = 1
 ' Access Style Attributes
StyleBag1.StyleName(0) = "Selected"
StyleBag1.StyleFlags(0) = BRD_BORDER + BRD_BEVEL
StyleBag1.StyleWidth(0) = 300
StyleBag1.StyleHeight(0) = 40
 ' The Font Attributes of the Style
StyleBag1.StyleFontName(0) = "Arial"
StyleBag1.StyleFontSize(0) = 10
StyleBag1.StyleFontBold(0) = True
StyleBag1.StyleFontItalic(0) = False
StyleBag1.StyleFontUnderline(0) = True
StyleBag1.StyleFontStrikeThru(0) = False
' The Color Attributes of the Style
StyleBag1.StyleForeColor(0) = cRed
StyleBag1.StyleBackColor(0) = cYellow
StyleBag1.StyleBorderColor(0) = cLtGrey
StyleBag1.StyleBevelLight(0) = cWhite
```

```
StyleBag1.StyleBevelDark(0) = cDkGrey
' The Other Display Attributes of the Style
StyleBag1.StyleBorderWidth(0) = 2
StyleBag1.StyleBevelInner(0) = 1
StyleBag1.StyleBevelOuter(0) = 1
StyleBag1.StyleShadowWidth(0) = 5
' Set Up Default Item Information:  Selected, Current, etc.
ctrl.StyleBag = StyleBag.hWnd
ctrl.DefStyle(DEFITEM_SELECTED) = "Selected"
ctrl.DefStyle(DEFITEM_CURRENT) = "Selected"
```

Property Access :

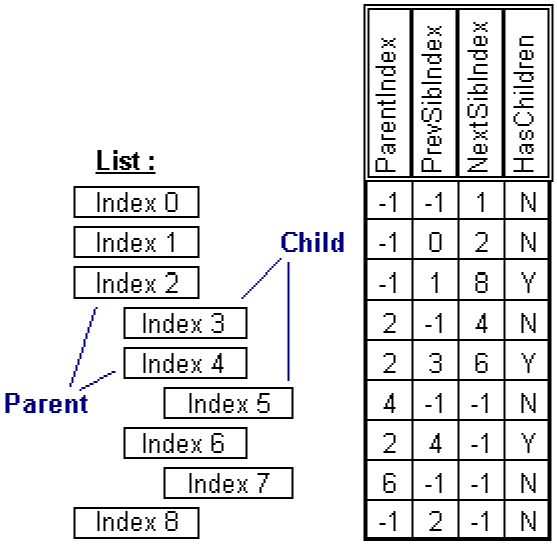
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The StyleBevelDark Property

See also :

Style Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

Long (Color), Property Array

Purpose :

determines the Color used for Sunken Bevelling of a defined Style that may be applied to an item.

Description :

This property specifies the color to use when drawing the three-d bevelling around an item.
For the three-d effects, the light source is always assumed to come from the top-left corner of the screen.
Three-d elements which are raised up on the top and left sides are colored with the BevelLight color. The same color is used for elements which are lowered on the bottom and right sides.
Three-d elements which are lowered on the top and left sides are colored with the BevelDark color. The same color is used for elements which are raised on the bottom and right sides.
Any width left over after the bevelling will be painted with the BorderColor.

Note that the style items BRD_BORDER and BRD_BEVEL flags must be set in the StyleFlags property to show the border and bevelling at all.

Example :

```
See example in StyleBag Property
```

Property Access :

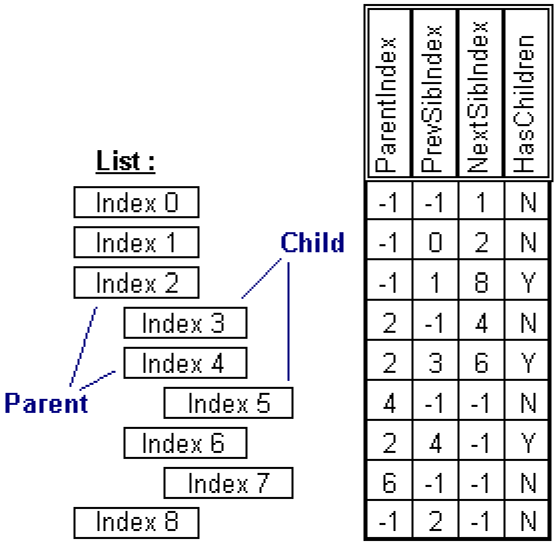
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleBevelInner Property

See also :

Style Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

Integer, Property Array

Purpose :

determines the Amount to Bevel Inside edge of Border of a defined Style that may be applied to an item.

Description :

The BevelInner Property allows you to specify the width of the bevelling to use on the inside edge of the control's border. A negative value will cause the inside edge of the border as lowered, a positive value displays as rasied.

Note that the style items BRD_BORDER and BRD_BEVEL flags must be set in the StyleFlags property to show the border and bevelling at all.

Example :

`See example in StyleBag Property`

Property Access :

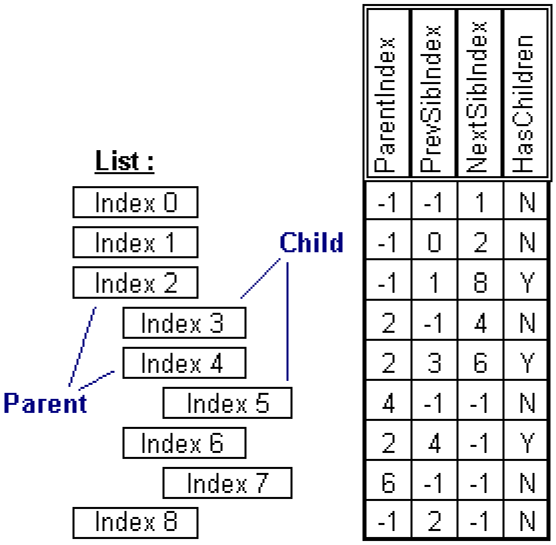
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleBevelLight Property

See also :

Style Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

Long (Color), Property Array

Purpose :

determines the Color used for Raised Bevelling of a defined Style that may be applied to an item.

Description :

This property specifies the color to use when drawing the three-d bevelling around an item.
For the three-d effects, the light source is always assumed to come from the top-left corner of the screen.

Three-d elements which are raised up on the top and left sides are colored with the BevelLight color. The same color is used for elements which are lowered on the bottom and right sides.
Three-d elements which are lowered on the top and left sides are colored with the BevelDark color. The same color is used for elements which are raised on the bottom and right sides.
Any width left over after the bevelling will be painted with the BorderColor.
Note that the style items BRD_BORDER and BRD_BEVEL flags must be set in the StyleFlags property to show the border and bevelling at all.

Example :

```
See example in StyleBag Property
```

Property Access :

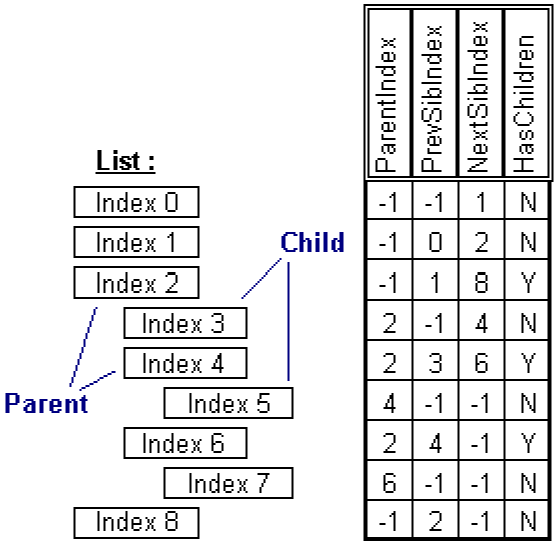
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleBevelOuter Property

See also :

Style Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

Integer, Property Array

Purpose :

determines the Amount to Bevel Outside edge of Border of a defined Style that may be applied to an item.

Description :

The BevelOuter Property allows you to specify the width of the bevelling to use on the outside edge of the control's border. A negative value will cause the outside edge of the border as lowered, a positive value displays as rasied.

Note that the style items BRD_BORDER and BRD_BEVEL flags must be set in the StyleFlags property to show the border and bevelling at all.

Example :

```
See example in StyleBag Property
```

Property Access :

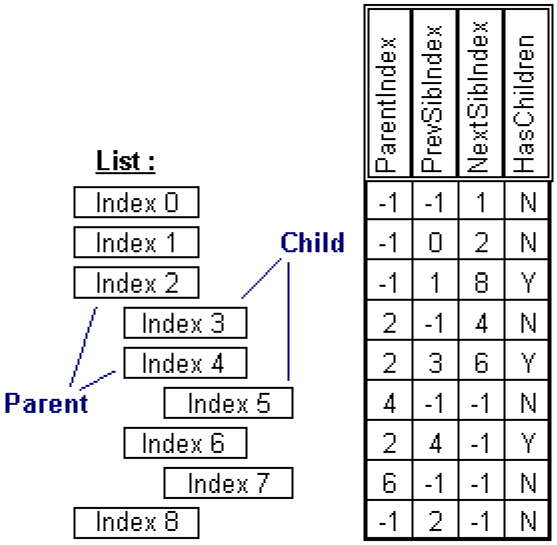
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleBold Property

See also :

Style Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

Boolean, Property Array

Purpose :

determines if the font is bold of a defined Style that may be applied to an item.

Description :

Each item in a StyleBag has a StyleName, StyleFontName, StyleFontSize, StyleBold, StyleItalic, StyleUnderline, StyleForColor, StyleBackColor, etc.

Example :

```
See example in StyleBag Property
```

Property Access :

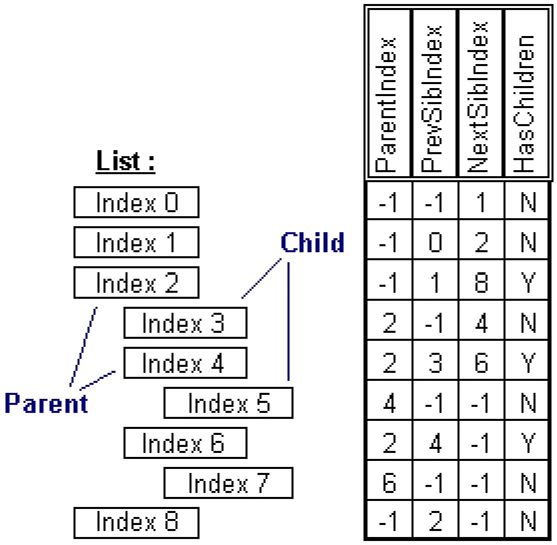
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleBorderColor Property

See also :

Style Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

Long (Color), Property Array

Purpose :

determines the Border Color of a defined Style that may be applied to an item.

Description :

Each item in a StyleBag has a StyleName, StyleFontName, StyleFontSize, StyleBold, StyleItalic, StyleUnderline, StyleForColor, StyleBackColor, etc.   See also Style Properties

Example :

`See example in StyleBag Property`

Property Access :

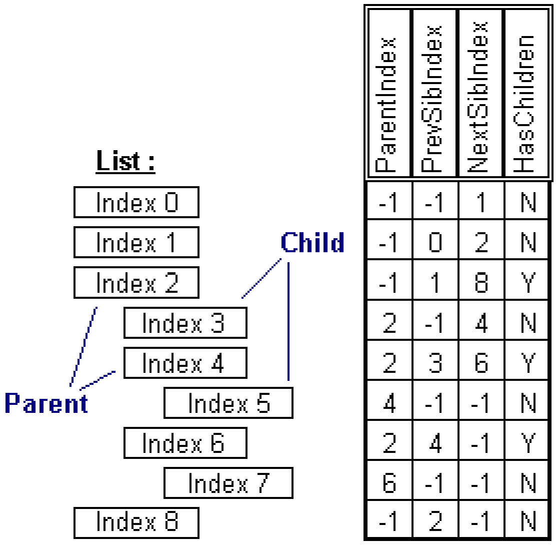
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleBorderWidth Property

See also :

Style Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

Integer, Property Array

Purpose :

determines the Border Width, in pixes, of a defined Style that may be applied to an item.

Description :

Each item in a StyleBag has a StyleName, StyleFontName, StyleFontSize, StyleBold, StyleItalic, StyleUnderline, StyleForColor, StyleBackColor, etc.

Example :

```
See example in StyleBag Property
```

Property Access :

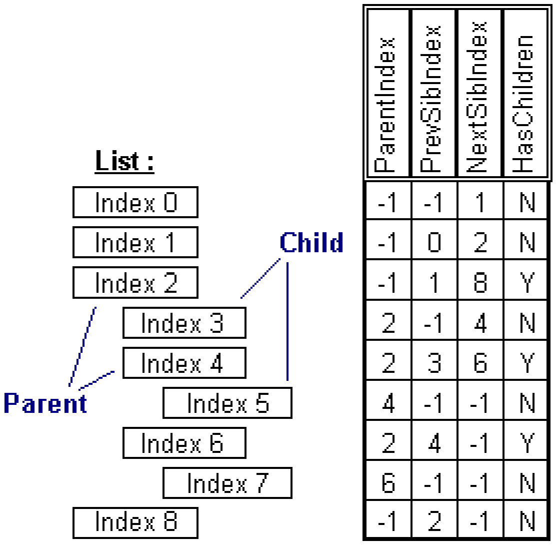
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleFlags Property

See also :

Style Properties
Item Flags String and Long Values
Codes and Constants

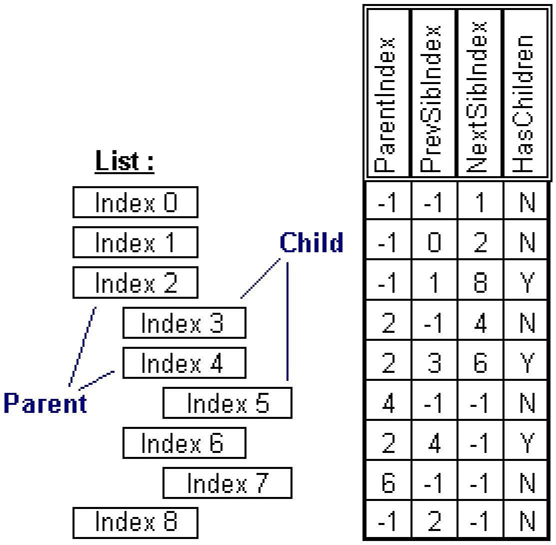
Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

String, Property Array

Purpose :

determines special flags for display and behaviour of a defined Style that may be applied to an item.

Description :

This property allows you to dynamically alter the display and behaviour of a style easily and efficiently. Using flags (either in string or numeric form) you are able to change the following aspects of a styles operation : Border Display, Text Placement, Highlight Styles, etc.

This property accepts and returns a string value; However, the string values are translated to and from long integer values internally. When using Basic, you can use either strings or longs when manipulating the contents of this property, the values will be converted appropriately.   Note that when using long integers with this property, it is very much like using the flags in the MsgBox function.

Setting the value of this property is a little different than when setting most other properties.   The most powerful way to set the ControlFlags, is to build a string flags to set or unset.   Flags must be seperated by a plus '+' or a minus '-'.   When preceeded by a plus, the flag will by set within the control.   Conversely, a minus preceeding a flag means that flag will be turned off within the control.   When using the string method, The new flags will be added to the current flags to arrive at new settings.   This is cool because you can customize certain aspects of the control, and make sure of some of the other settiings at any time.   If you use the long integer method, the current flags will be totally replaced by the new flags.   The same effect can be forced with the string method by omitting the first plus or minus.

Example :

```
See example in StyleBag Property
```

Property Access :

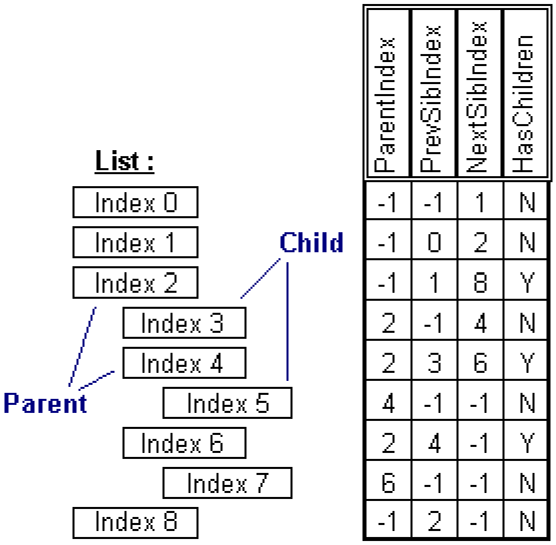
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleFontName Property

See also :

Style Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

String, Property Array

Purpose :

determines the Font Name of a defined Style that may be applied to an item.

Example :

```
See example in StyleBag Property
```

Property Access :

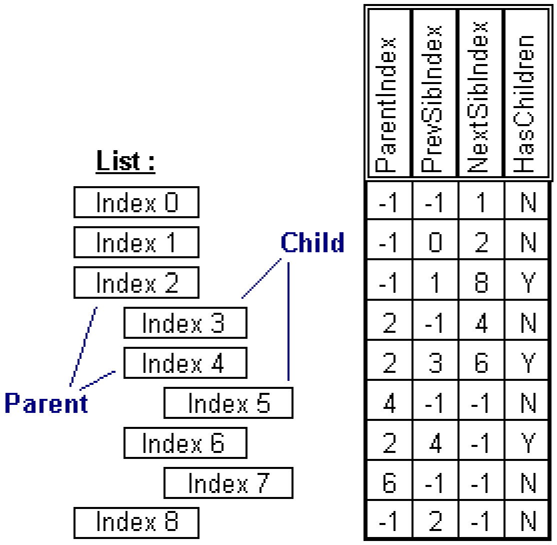
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleFontSize Property

See also :

Style Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

Single, Property Array

Purpose :

determines the Font Size of a defined Style that may be applied to an item.

Example :

```
See example in StyleBag Property
```

Property Access :

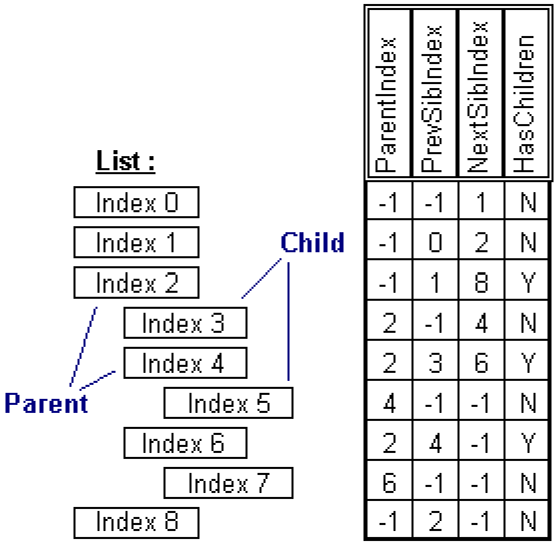
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleForeColor Property

See also :

Style Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

Long (Color), Property Array

Purpose :

determines the Color to dis[play text of a defined Style that may be applied to an item.

Example :

`See example in StyleBag Property`

Property Access :

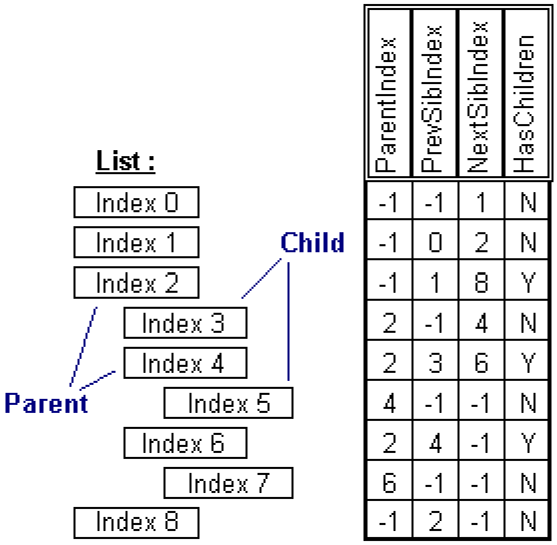
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleHeight Property

See also :

Style Properties
ItemHeight

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

Integer, Property Array

Purpose :

determines the height of a defined Style that may be applied to an item.

Description :

Specifies the width of an individual style this property is not set, the control will automatically calculate the item's size based on the text, style, and image.

Note that this property must be set in order to enable any word-wrapping set with the StyleFlags property.

Example :

```
See example in StyleBag Property
```

Property Access :

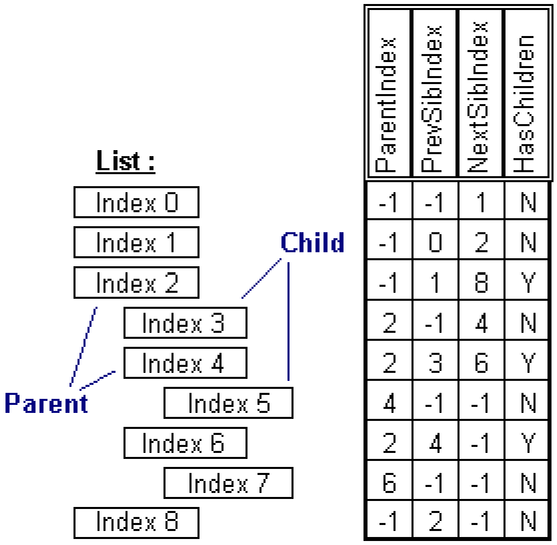
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleItalic Property

See also :

Style Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

Boolean, Property Array

Purpose :

determines if the Font used is Italic for a defined Style that may be applied to an item.

Example :

`See example in StyleBag Property`

Property Access :

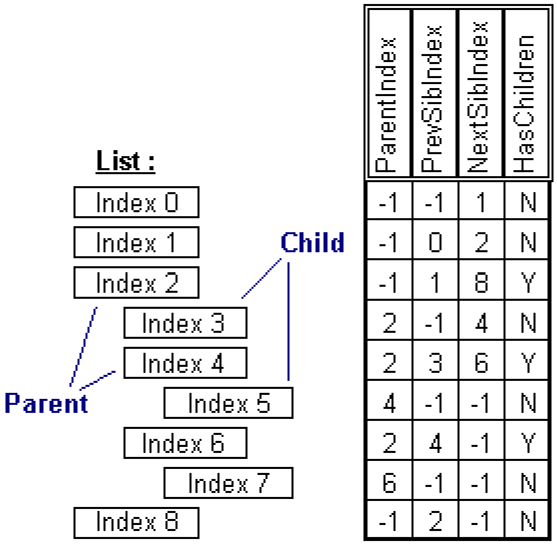
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleName Property

See also :

Style Properties
ImageName

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

String, Property Array

Purpose :

Make code more readable by referrencing Styles by name from the StyleBag.

Description :

Each item in the StyleBag can have a StyleName. StyleName can help you keep track of Styles in a better way. Perhaps you have a style set up that you want applied to all selected items. Instead of remembering that the selected style is at index 0, you can just assign the StyleName( ) to be Selected In code, you can then referrence it by name as in the example.

Example :

```
See example in StyleBag Property
```

Property Access :

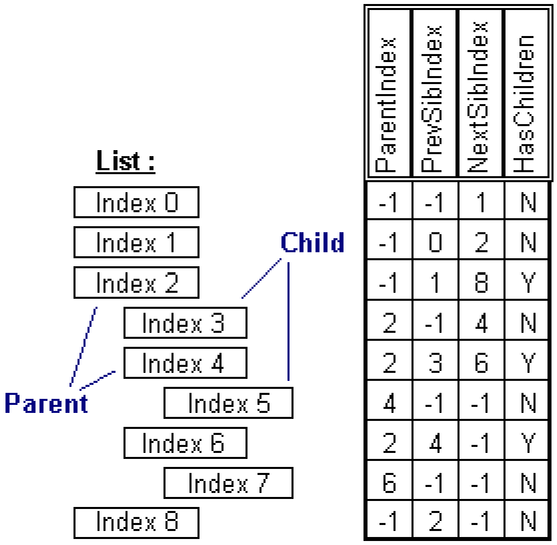
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleShadowWidth Property

See also :

Style Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

Integer, Property Array

Purpose :

determines if the Width of Shadow for a defined Style that may be applied to an item.

Description :

The StyleShadowWidth Property allows you to specify the width of the shadow surrounding an item.   This width is applied to the right and bottom of the item.

Note that the BRD_SHADOW flag must be set in the StyleFlags to show the shadow at all.

Example :

`See example in StyleBag Property`

Property Access :

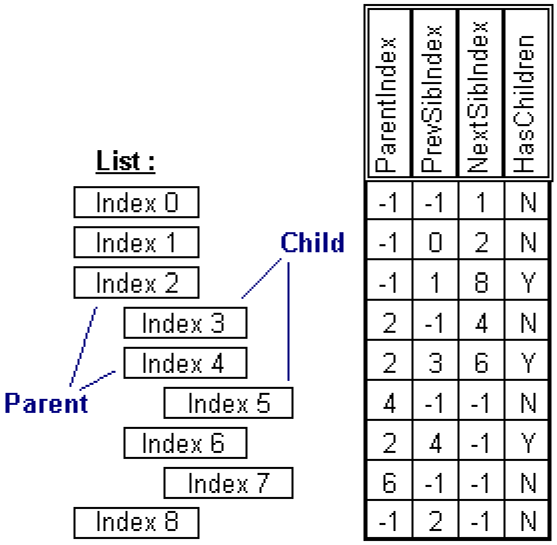
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleStrikethru Property

See also :

[Style Properties]

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

Boolean, Property Array

Purpose :

determines if the Font used has StrikeThru for a defined Style that may be applied to an item.

Example :

`See example in StyleBag Property`

Property Access :

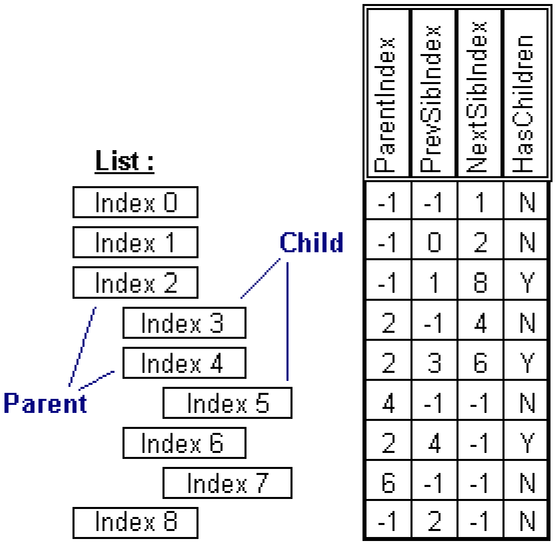
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleUnderline Property

See also :

Style Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

Boolean, Property Array

Purpose :

determines if the Font used has an Underline for a defined Style that may be applied to an item.

Example :

```
See example in StyleBag Property
```

Property Access :

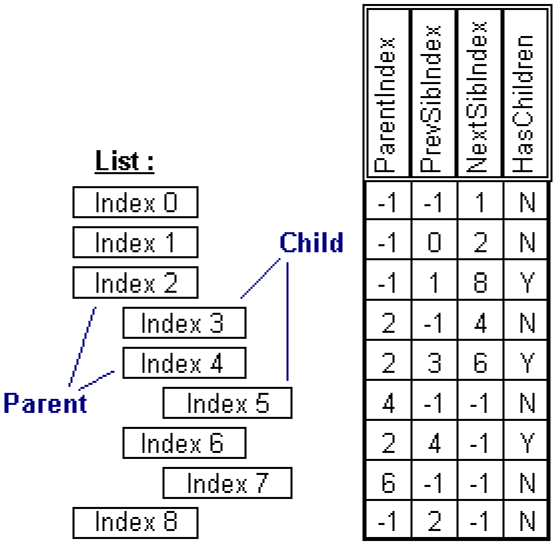
Property Window : No
Des Time : none
Run Time : Read-Write

The StyleWidth Property

See also :

Style Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag

Property Type :

Integer, Property Array

Purpose :

determines the width of a style, in pixels, for a defined Style that may be applied to an item.

Example :

```
See example in StyleBag Property
```

Property Access :

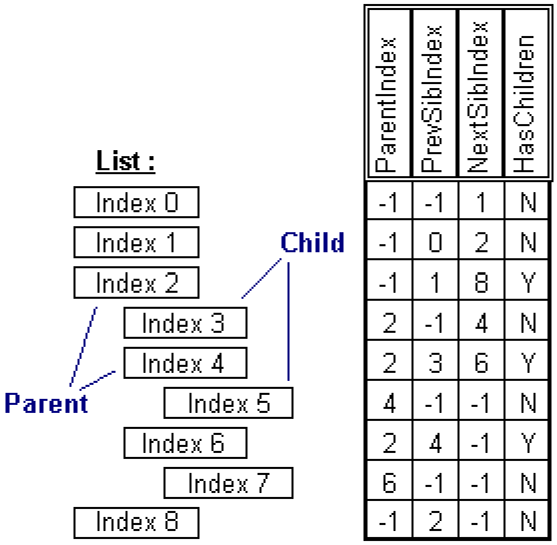
Property Window : No
Des Time : none
Run Time : Read-Write

The TitleText Property

See also :

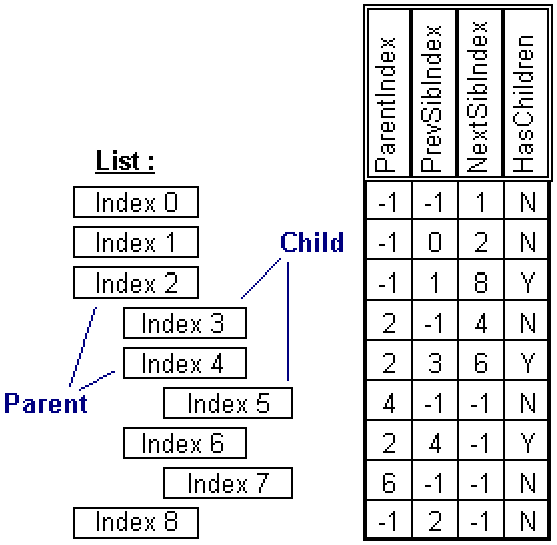
List Display Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList

Property Type :

String

Purpose :

text displayed at top of control

Description :

This property specifies a formatted text string that will be displayed at the top of the control.

Example :

```
- Setting Values with Code :
Dim S As String
S = "This is a List Cool Things"
```

```
Ctree1.TitleText = S
```

Property Access :

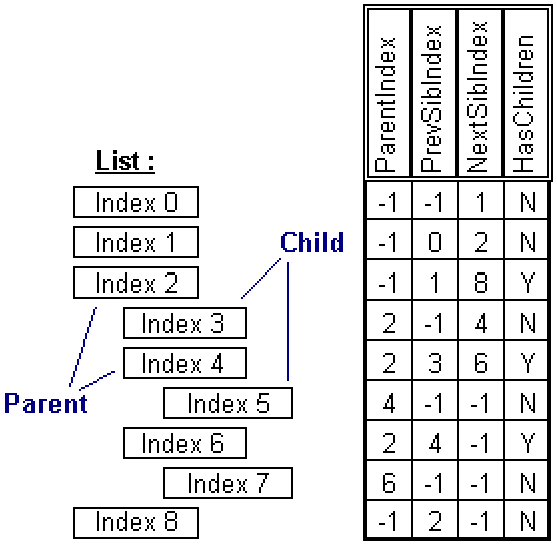
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The TokenNum Property

See also :

Parsing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse

Property Type :

Integer, Property Array

Purpose :

identify exactly which token was found

Description :

The TokenNum Property returns the Index of a Token as it is found in the Symbols or Keywords List; Furthermore, this property only applies to Tokens which are of TokenType Symbol or Keyword (this property will return -1 otherwise).
This property is useful to determine exactly which Symbol or Keyword a particular Token is.   This allows you to treat different Symbols and Keywords differently.

Example :

```
- Getting Values with Code :
Dim I As Integer
' Force Keyword #4 to UpperCase
For I = 1 to (LParse.ListCount - 1)
    If (LParse.TokenType(I) = 1) Then
        If (LParse.TokenNum(I) = 3) Then
            LParse.List(I) = UCase$(LParse.List(I))
        End If
    End If
Next I
```

Property Access :

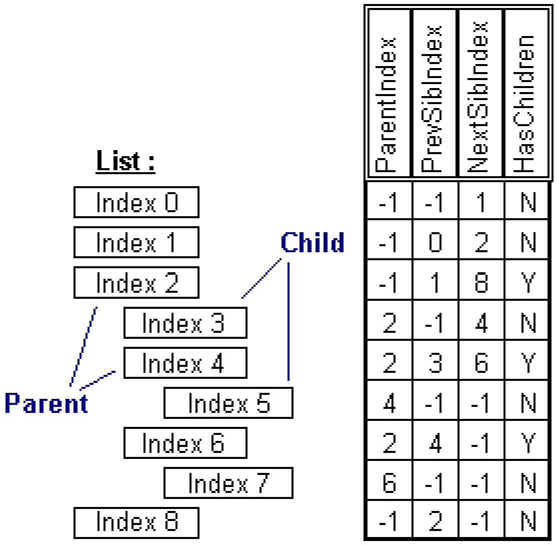
Property Window : No
Des Time : none
Run Time : Read-Write

The TokenType Property

See also :

Parsing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse

Property Type :

Integer, Property Array

Purpose :

identifies type of token (identifier, keyword, etc.)

Description :

The TokenType Property returns a number, specifying how a particular Token was parsed out of the ParseString. The Values for this property are as follows :

0 - Identifier
1 - Keyword
2 - Symbol
3 - Whitespace
4 - Quoted String
5 - Numeric

As Tokens are Parsed out of the ParseString and added to the Tokens List, each Token will be parsed out as one of the above types. The Quoted String and Numeric types will only be found if the appropriate b flags are set. The ParseString is checked for tokens in the following order : WhiteSpace, Quotes, Numerics, Symbols, Keywords. If none of these can be found, then the Token is an Identifier and includes all characters up to the next whitespace or symbol.

Example :

```
- Getting Values with Code :
Dim I As Integer
' Change All Keyword Tokens to UpperCase
For I = 0 To (LParse.ListCount - 1)
    If LParse.TokenType(I) = 1 Then
        LParse.List(I) = UCase$(LParse.List(I))
    End If
Next I
```

Property Access :

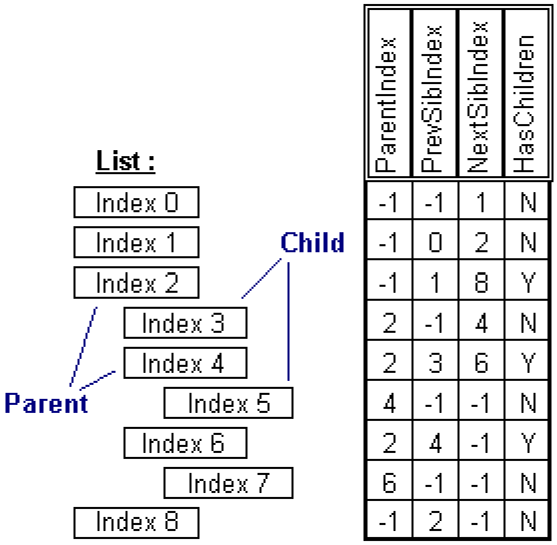
Property Window : No
Des Time : none
Run Time : Read-Only

The TopIndex Property

See also :

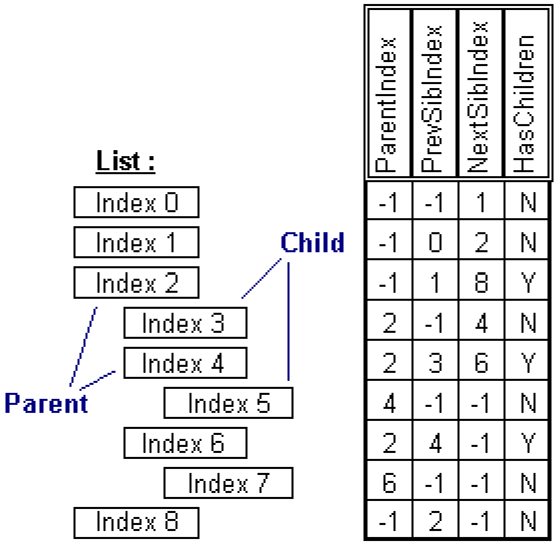
List Data Properties

Applies To :



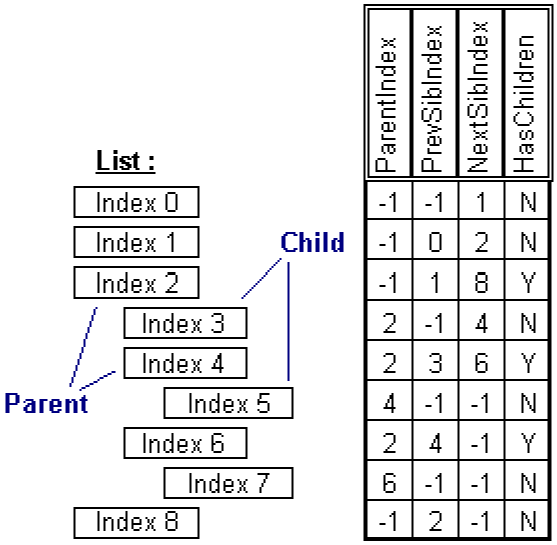
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

CTree



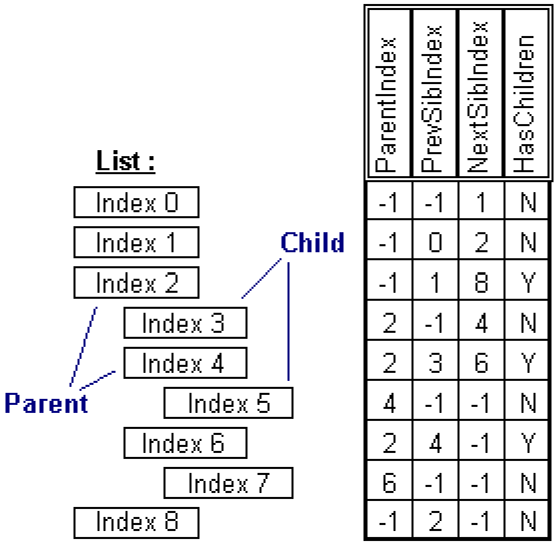
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

EnhList



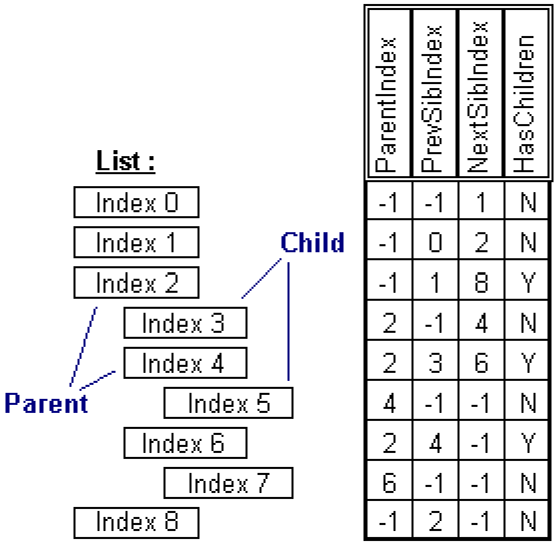
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



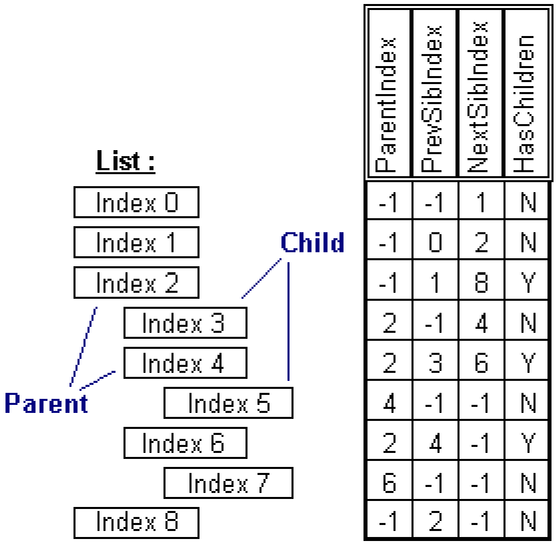
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Long

Purpose :

First Visible Item in List Display

Description :

The TopIndex Property indicates the Index of the Item which is the topmost visible Item in the listbox. This value will automatically change when the listbox is being scrolled.

Note: This property is useful to programmatically scroll the listbox up and down.

Example :

```
- Setting Values with Code :
' Scrolls Top of List
List.TopIndex =  0
' Scroll to Currently Selected Item
List.TopIndex = List.ListIndex
' Scroll to End of List
List.TopIndex = List.ListCount
```

```
- Getting Values with Code :
Dim I As Integer
I = List.TopIndex
```

Property Access :

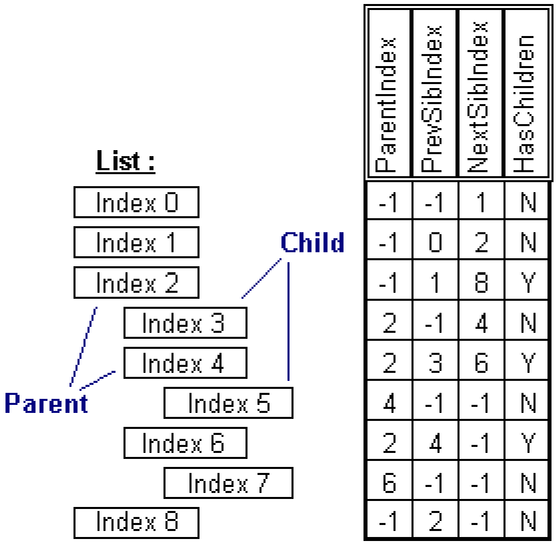
Property Window : No
Des Time : none
Run Time : Read-Write

The TrueShadow Property

See also :

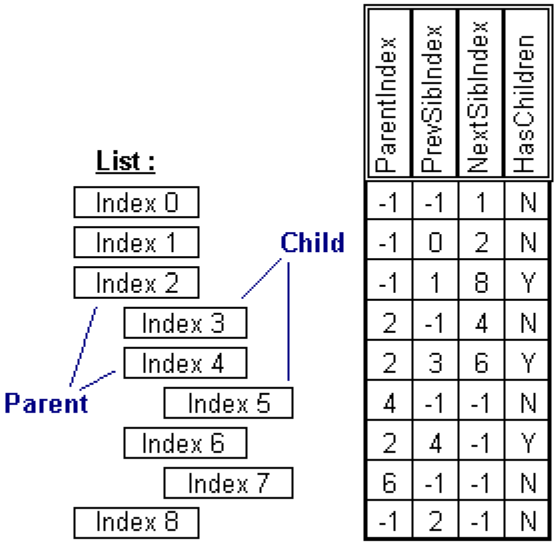
Control Border Properties

Applies To :



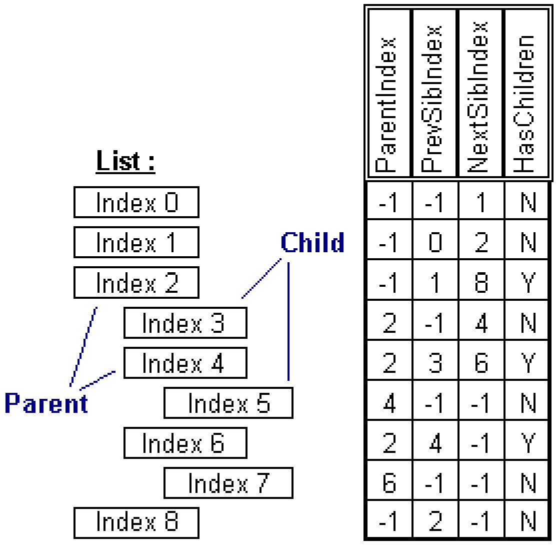
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

ImageBag



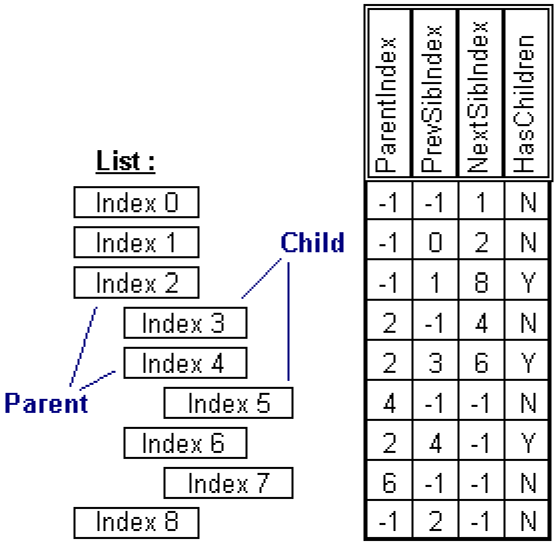
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

StyleBag



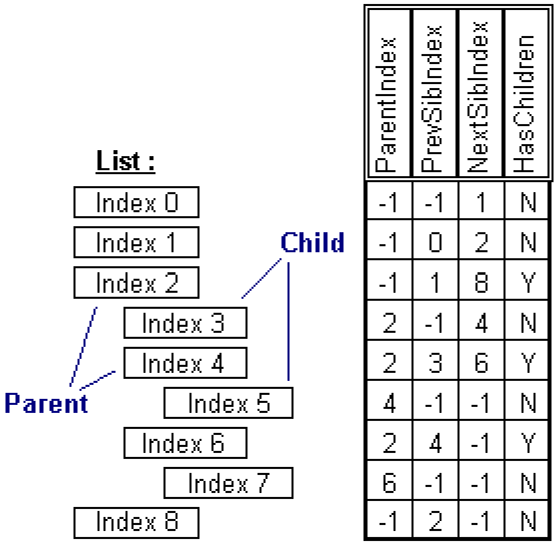
ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TabRas



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSlabel



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

TSTree

Property Type :

Boolean

Purpose :

set to draw shadow with a transparent background

Description :

This property lets you have a shadow on your control which lets the current background bleed through. This feature heightens the effect of the shadow immensly. This property is best used when using the popup mode (see the PopUp Property).

Note that a control's BRD_SHADOW flag must be set in the ControlFlags Property to make the shadow visible at all.

Example :

```
- Setting Values with Code :
ctrl.TrueShadow = true
```

Property Access :

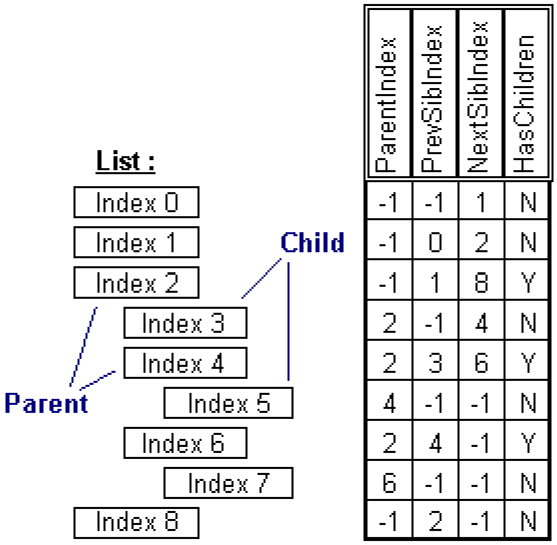
Property Window : Yes
Des Time : Read-Write, Saved
Run Time : Read-Write

The WSpaceChars Property

See also :

Parsing Properties

Applies To :



ParentIndex	PrevSibIndex	NextSibIndex	HasChildren
-1	-1	1	N
-1	0	2	N
-1	1	8	Y
2	-1	4	N
2	3	6	Y
4	-1	-1	N
2	4	-1	Y
6	-1	-1	N
-1	2	-1	N

LParse

Property Type :

String

Purpose :

identifies Whitespace (i.e. blank) characters

Description :

This Property along with the Symbols List, provides for the primary function of LParse. That is, Identifying Tokens within the ParseString and listing them as individual Items in the Parsed String List. Typically, the WSpaceChars would contain at least some of the following characters : space, tab, carraige-return, and linefeed.

The WSpaceChars Property is a string containing characters which are used, during the Parsing Operation, to Identify and delimit Tokens found in the ParseString Property. Whitespace refers to characters which serve no real purpose other than to separate words or tokens.   Whitespace characters are similar to Symbols in that they are both used to determine the end of a particular token; However, they differ in that symbols may contain more than one character.   LParse also treats whitespace a little differently than Symbols when copying it to the Parsed String List.   If a series of whitespace characters are found together within the ParseString, such as three spaces in a row, all the characters will be copied to the Parsed String List as a single Item. In the case of Symbols, each Symbol encountered is copied are added to the list as separate items.

Example :

```
- Setting Values with Code :
Dim S As String
S = " "       ' Space Char
S = Chr$(8)   ' Tab Char
S = Chr$(13)  ' Carraige-Return Char
S = Chr$(10)  ' Linefeed Char
LParse.WSpaceChars = S

- Getting Values with Code :
Dim S As String
S = LParse.WSpaceChars
```

Property Access :

Property Window : No
Des Time : Read-Write, Saved
Run Time : Read-Write