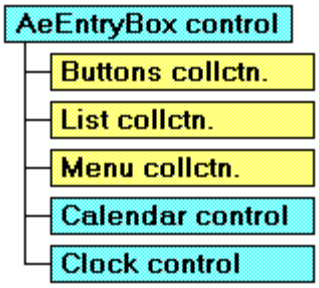
AeEntryBox Control

Members

Properties
Methods
Events

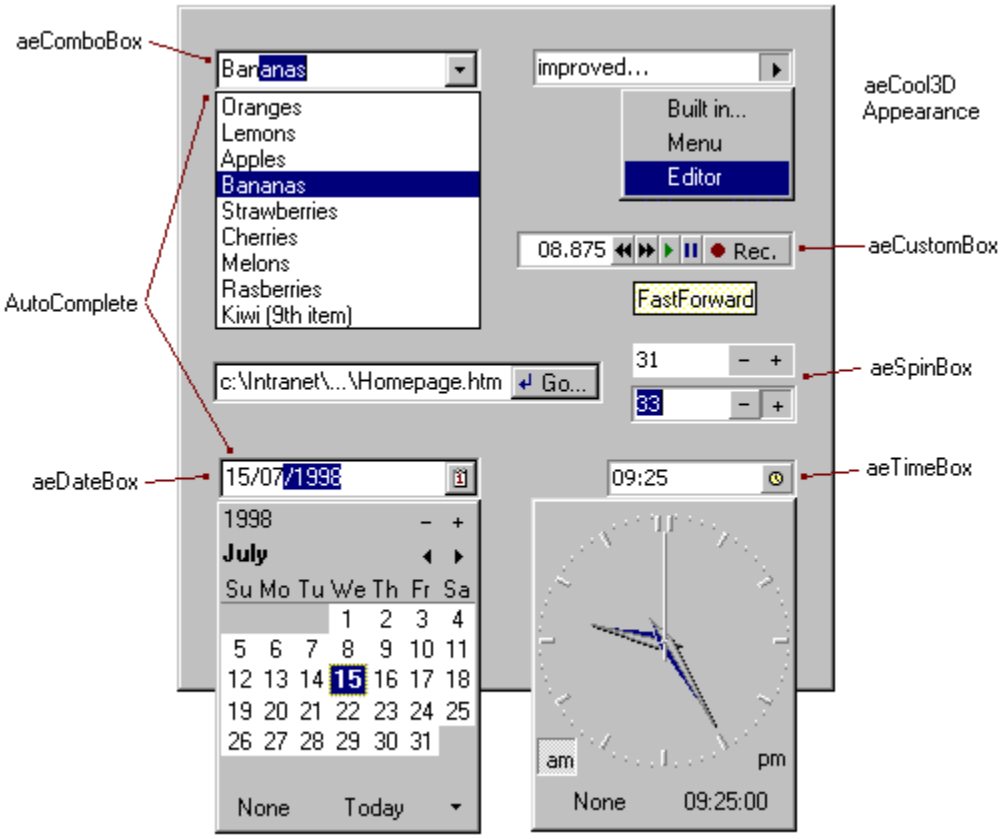


For most requirements, simply setting the Style property to 1 of 7 types of data entry, will be sufficient. This will configure the control's buttons and properties to defaults for that type of data. All the styles are based on a text area on the left, and one or more buttons on the right, aiding the user enter the required text.

In addition to the predefined styles, the combo buttons, drop-down list and popup menu can all be configured by using the property pages at design-time, or using the exposed object model at run-time. (See the Buttons, List and Menu objects for more details)

Other new features include:

+ With AutoComplete on, selected text is appended as the user types by searching the list for matching entries, or using the FormatString and DataType properties to construct default values for date and numeric types.
+ Styles can be changed at run-time, for building forms "on the fly".
+ UpperCase and AutoCapsLock properties for handling text case.
+ Simplify code with automatic conversion and validation using the Value property, which reads/writes a variant value, and the IsValid method, which validates the text, both according to the DataType property.
+ EnterKeyBehaviour allows users to tab to the next control, or click the first button.



Note The AeEntryBox.List property provides a **AeList** object with it's own methods and properties for managing the list items. This differs from a standard ComboBox control, which provides the Listxxx methods and properties.

AeEntryBox Properties

Alignment	MaxValue/MinValue
AllowEdit	Menu
Appearance	MouseIcon
BorderStyle	MousePointer
AutoCapsLock	Name
AutoComplete	Object
Buttons	Parent
BackColor	PasswordChar
Container	Picture
DataField	PictureWidth
DataSource	PopupForm
DataType	SelLength
DragIcon	SelStart
DragMode	SelText
Enabled	Step
EnterKeyBehaviour	Style
Font	TabIndex
ForeColor	TabStop
FormatString	Tag
Height	Text
HelpContextID	ToolTipText
hWnd	Top
Index	UpperCase
Left	UseMaskColor
List	Value
Locked	Visible
MaskColor	WhatsThisHelpID
MaxLength	Width

AeEntryBox Methods

Action
CreatePopupForm
Drag
IsValid
Move
Refresh
SetFocus
ShowWhatsThis
ZOrder

AeEntryBox Events

ActionSet
ButtonClick
Change
Click
DblClick
DragDrop
DragOver
GotFocus
KeyDown
KeyPress
KeyUp
LostFocus
MenuClick
MenuHighlight
MenuPopup
MouseDown
MouseMove
MouseUp

Style Property

Returns or sets the type data entry the AeEntryBox control is used for.

Syntax *Object*.**Style** [= *Value*]

Part	Description
Object	An **AeEntryBox** control.
Value	An AeEntryBoxStyles constant.

Settings The settings for **AeEntryBoxStyles** are

Setting	Description
0 - aeCustomBox	Custom built styles.
1 - aeTextBox	Standard TextBox. DataType is set to vbString, buttons are cleared.
2 - aeComboBox	Standard ComboBox. DataType is set to vbString, and a button with aePopupList action added.
3 - aeSpinBox	Improved Spin controls. DataType is set to vbLong, and a pair of buttons with aeIn/DecreaseValue actions, and RepeatRates set for spinning when the mouse is held down, added.
4 - aeDateBox	Advanced Date entry. DataType is set to vbDate, and a button with aePopupCalendar action added.
5 - aeTimeBox	Advanced Time entry. DataType is set to vbDate, and a button with aePopupClock action added.
6 - aeYesNoBox	For when a CheckBox doesn't fit. DataType set to vbBoolean, FormatString set to "Yes/No", pair of aeGroup buttons with aeSetTrue/aeSetFalse actions added.
7 - aeNavigationBox	Unbound DataControl. DataType set to vbString, six buttons added (4 navigation symbols, add and delete), with no actions set.

Remarks The Style property provides a quick and simple way to set up an AeEntryBox for standard types of data. Whenever the Style is changed (at design or run-time), the DataType, FormatString and Value properties, and Buttons collection, are adjusted to pre-defined settings that best suit the type of data entry required.

In most development situations, the Style will be the first property set, according to the type of data entry required. Subsequent changes can then be made to refine the behaviour of the control.

Whenever the **DataType**, **Value**, **FormatString** or **Buttons** members are changed after the Style has been set, the Style is reset to aeCustomBox.

Tip Experimenting with the different Style settings is a good way to learn the capabilities of the AeEntryBox control.

Value Property

Returns or sets a variant value of the text in an AeEntryBox Control.

Syntax *Object*.**Value** [= *NewValue*]

Part	Description
Object	An AeEntryBox control.
NewValue	A Variant expression.

Remarks The Value property works in conjunction with the DataType and FormatString properties to provide convenient conversion of the text entered in an AeEntryBox control.

When reading the Value property, the text entered is converted into the appropriate Variant value according to the **DataType**. When setting the Value the **FormatString** is used to create the text entry in the same way as the VB **Format** function.

If the Text entered can not be converted into the specified data type, an Empty value is returned. The IsValid method can also be used.

DataType Property

Returns or sets the type of data of the text entered into an AeEntryBox control.

Syntax *Object*.**DataType** [= *Value*]

Part	Description
Object	An AeEntryBox control.
Value	An AeDataTypes constant.

Settings The settings for **AeDataTypes** are:

Setting	Description
2 - aeInteger	Whole numbers.
3 - aeLong	Large whole numbers.
4 - aeSingle	Single precision numbers.
5 - aeDouble	Double precision numbers.
6 - aeCurrency	Currency values. FormatString set to "Currency". Formatted as decimal when the control is active.
7 - aeDate	FormatString set to "Short Date". AutoComplete appends today's date.
8 - aeString	Text entries.
11 - aeBoolean	True or False. FormatString set to "Yes/No". AutoComplete enters Yes or No.
12 - aeVariant	Variable contents.
13 - aeTime	FormatString set to "Time". AutoComplete appends current time.
14 - aeDecimal	Decimal numbers.
17 - aeByte	Single character.

Remarks The DataType determines what type of text the AeEntryBox should display, ie numbers, names, currency values etc..

When the DataType is changed the FormatString is cleared or set to one of the presets listed in the table above. You can overide the default setting by subsequently changing the FormatString.

The AeDataType constants are compatible with the **VbVarType** constants in Visual Basic except for aeTime.

The Value property uses the DataType to determine which conversion to apply to the text entered, and the control uses the DataType for various other operations such as the Action settings. Therefore, if the DataType has not been set correctly, unexpected behaviour may result.

FormatString Property

Returns or sets the format used to display text in an AeEntryBox control.

Syntax *Object*.**FormatString** [= *Value*]

Part	Description
Object	An AeEntryBox control.
Value	A String expression specifying the format string.

Remarks The FormatString is used to convert values in to text using the same formatting as the **Format$** function in Visual Basic. Please refer to the Visual Basic help for a full listing of the format codes available.

Setting the DataType property, automatically sets the FormatString to preset default values appropriate for that data type. In most situations the default FormatStrings will be sufficient. Custom FormatStrings, however can be entered for specific requirements.

The system formats for date, time and currency are used for international compatibility.

UpperCase Property

Returns or sets whether text typed into an AeEntryBox control is converted into upper case.

Syntax *object*.**UpperCase** [= *Value*]

Part	Description
object	An AeEntryBox control.
Value	A Boolean expression.

Remarks When the **UpperCase** is set to True, all characters typed in the control will be forced into upper case. The default is False.

AllowEdit Property

Returns or sets whether the user can edit the text in an AeEntryBox control.

Syntax *Object*.**AllowEdit** [= *Value*]

Part	Description
Object	An AeEntryBox control.
Value	A Boolean expression.

Remarks When **AllowEdit** is set to False, the user will not be able to edit the text portion of the control directly with a text cursor. When the control becomes active, the text becomes highlighted with a focus rectangle. Clicking the text area also pops up the list, if available. This behaviour is functionally compatible to a standard **ComboBox** control with the Style property set to vbComboDropDownList.

When **AllowEdit** is set to True (default), the text area of the control behaves like a standard **TextBox.**

EnterKeyBehaviour Property

Returns or sets the behaviour of an AeEntryBox control when the Enter key is pressed.

Syntax *Object*.**EnterKeyBehaviour** [= *New_EnterKeyBehaviour*]

Part	Description
Object	An **AeEntryBox** control.
New_EnterKeyBehaviour	A AeEnterKeyBehaviours constant, defining the response to the Enter key.

Settings The settings for **AeEnterKeyBehaviours** are:

Setting	Description
0 - aeDefaultEnterKey	The enter key is handled like a standard TextBox.
1 - aeForwardFocus	Focus is moved to the next control in the tab order when the enter key is pressed. This behaviour is equivalent to pressing the Tab key.
2 - aeCustomEnterKey	The Action of the first button is performed when the enter key is pressed.

Remarks The extra responses to the enter key are provided to allow developers to create screens with faster and more natural data entry.

The standard Windows behaviour is to employ default CommandButtons, however this is frequently confusing for users. The aeForwardFocus can be used to provide a more traditional data entry screen, and the aeCustomEnterKey can be used for custom behaviour (also prevents Windows from beeping).

Note: If the control is on a form that includes a CommandButton with it's Default property set to True, the control will not receive the enter key stroke, regardless of the EnterKeyBehaviour setting.

AutoComplete Property

Returns or sets whether text is automatically completed as the user types it into an AeEntryBox control.

Syntax *Object*.**AutoComplete** [= *Value*]

Part	Description
Object	An AeEntryBox control.
Value	A Boolean expression.

Remarks When **AutoComplete** is set to True (default), the control will intelligently guess what the user is about to type, and automatically complete the entry with selected text.

A combination of other properties are used to complete the entries. If the DataType is set to aeDate or aeTime, the current date or time respectively, are completed. If the List contains any items, it is searched and the best match is completed.

This particular implementation greatly increases productivity speed and, unlike other methods such as "masked" entries, maintains a simple and consistent user-interface.

MaxValue, MinValue Properties

Returns or sets the maximum and minimum values that can be entered into an AeEntryBox or AeCalendar control.

Syntax *Object*.**MaxValue** [= *New_Value*]

Object.**MinValue** [= *New_Value*]

Part	Description
object	An AeEntryBox or AeCalendar control.
New_Value	A Variant expression that can be converted to the AeEntryBox's DataType property, or a Date for the AeCalendar.

Remarks Controls with a numeric DataType can be limited to a range of values with **MaxValue** and **MinValue** properties.

The limits set are applied in the following situations:

- When aeIncrement/aeDecrement Actions are performed, such as by pressing the plus/minus keys or clicking a button set to change the value.
- When the Value is set to a value outside the range.
- When the IsValid property is inspected.

The popup calendar assumes the Min/MaxValues of the AeEntryBox control.

Action Method

Performs an action on an AeEntryBox control.

Syntax *Object*.**Action** *ActionId*

Part	Description
Object	An AeEntryBox control.
ActionId	An AeEntryBoxActions constant specifying the action to be performed.

Settings The settings for **AeEntryBoxActions** are:

Setting	Description
0 - aeCustomAction	No action is performed.
1 - aeIncreaseValue	The value for numeric data types is increased by value of the Step property.
2 - aeDecreaseValue	As aeIncrement, but decreases the value.
3 - aePopupList	Shows/Hides the popup list.
4 - aePopupCalendar	Shows/Hides the popup calendar.
5 - aePopupClock	Shows/Hides the popup clock.
6 - aePopupMenu	Shows/Hides the popup menu.
7 - aePopupCustom	Shows/Hides the custom popup form.
8 - aeSetTrue	Sets the value to True.
9 - aeSetFalse	Sets the value to False.

Remarks The actions are also assigned to each button, using the Buttons property pages or Buttons.Actions property. These actions are performed whenever the button is clicked. providing a convenient way of setting up the control for common operations.

The ActionSet event is triggered whenever the Action method is called.

The CreatePopupForm method must be called first before using the aePopupCustom action.

If a popup form is already visible, performing one of the popup actions again will hide it.

ButtonClick Event

Occurs whenever a Button in an AeEntryBox control is clicked.

Syntax **Private Sub** *Object*_**ButtonClick**(*Button* **As Integer**)

Part	Description
Object	An AeEntryBox Control.
Button	The index of the button clicked.

Remarks The ButtonClick event can be used to design buttons that perform customised operations, for example, opening a search form to find the correct entry.

When the button is clicked, the button's Action will be performed on the AeEntryBox before the ButtonClick event is raised.

MaxLength Property

Returns or sets the maximum number of characters a user can enter into an AeEntryBox control.

Syntax *Object*.**MaxLength** [= *New_MaxLength*]

Part	Description
Object	An **AeEntryBox** control.
New_MaxLength	An Integer expression specifying the number characters.

Remarks A value of 0 (default) sets no limit.

IsValid Method

Returns True if the text of an AeEntryBox control can be converted to a variant value of the appropriate DataType. Read-only at run-time, not available at design-time.

Syntax *Object*.**IsValid**()

Part	Description
Object	An **AeEntryBox** Control.

Remarks Use this property in data entry forms where the user entry validation is required.

Blank entries return True. aeBoolean **DataType**s also return True regardless of the text entered, since the VB runtime offers limited boolean validation, and in most situations, boolean data entry is easily restricted, i.e. **AllowEdit** is False.

ActionSet Event

Occurs whenever the Action method is called for an AeEntryBox control.

Syntax **Private Sub** *Object*_**ActionSet**(*ActionCode* **As AeEntryBoxActions**, *Cancel* **As Boolean)**

Part	Description
Object	An **AeEntryBox** Control.
ActionCode	A AeEntryBoxActions constant specifying the action about to be performed.
Cancel	A Boolean value which is False. Setting this to True will cancel the action.

Settings The settings for **AeEntryBoxActions** are:

Setting	Description
0 - aeCustomAction	No action is performed.
1 - aeIncreaseValue	The value for numeric data types is increased by value of the Step property.
2 - aeDecreaseValue	As aeIncrement, but decreases the value.
3 - aePopupList	Shows the popup list.
4 - aePopupCalendar	Shows the popup calendar.
5 - aePopupClock	Shows the popup clock.
6 - aePopupMenu	Shows the popup menu.
7 - aePopupCustom	Shows the custom popup form.
8 - aeSetTrue	Sets the value to True.
9 - aeSetFalse	Sets the value to False.

Remarks

AutoCapsLock Property

Returns or sets whether the keyboards CapsLock is automatically turned off, when the shift key is used typing a letter into an AeEntryBox control.

Syntax *Object*.**AutoCapsLock** [= *Setting*]

Part	Description
Object	An **AeEntryBox** Control.
Setting	A Boolean expression

Remarks With AutoCapsLock set to True (default), when the user accidentally leaves the CapsLock on and then enters a letter with the shift key press, the CapsLock will be turned off, and an upper case letter entered.

The behaviour of the standard TextBox, is for a lower case letter to be inserted. Whilst this can be useful for entering long strings of upper case characters with occasional lower case ones, more often than not, the user just left the CapsLock on from a previous control.

CreatePopupForm Method

Creates a custom popup form for an AeEntryBox or AeCommandBox control. Returns a AePopupForm object.

Syntax Set *PopupForm* = *Object*.**CreatePopupForm**(*hWnd*)

Part	Description
PopupForm	A **AePopupForm** variable.
Object	An **AeEntryBox** or **AeCommandBox** control.
hWnd	A Long expression specifying the Window handle of the form or control to be displayed by the aePopupCustom Action.

Remarks The **AePopupForm** object allows you to hook into popup mechanism of the control and show your own forms and controls, providing a powerful way of expanding the functionality of the controls.

The topic Creating a Custom Popup Form, provides examples and more information on using the CreatePopupForm method.

Step Property

Returns or sets the incremental value applied when using the aeIncreaseValue and aeDecreaseValue actions on an AeEntryBox control.

Syntax *Object*.**Step** [= *Value*]

Part	Description
Object	An **AeEntryBox** Control.
Value	A Double expression specifying the incremental value.

Remarks The Step property is set to default values whenever the DataType is changed. You can change the default values, typically a aeSpinBox style control might need to be ‘spun’ 10 at a time rather than 1. aeDate and aeTime default to 1 day and 1 minute respectively.

Alignment Property

Returns or sets the text justification in an AeEntryBox control.

Syntax *Object*.**Alignment** [= *New_Alignment*]

Part	Description
Object	An **AeEntryBox** control.
New_Alignment	An AlignmentConstants value specifying the type of alignment.

Settings The settings for **AlignmentConstants** are:

Setting	Description
0 - vbLeftJustify	Text is aligned on the left side (default).
1 - vbRightJustify	Text is aligned on the right side.
2 - vbCenter	Text is aligned in the center.

Remarks The Alignment property only effects the text when control doesn't have the focus or the AllowEdit property is set to False. When the control text is being editing, the alignment is always on the left.

The Alignment property can also be changed at run-time.

Picture Property

Returns or sets the picture displayed to the left of the AeEntryBox Control.

Syntax **Set** *Object*.**Picture** [= *New_Picture*]

Part	Description
Object	A **AeEntryBox** control.
New_Picture	A valid Picture expression. Windows bitmap and icons supported.

Remarks Use the PictureWidth property to set the amount of space allocated for picture.

PictureWidth Property

Returns or sets the width in pixels of the space available for pictures or indentation, in a AeEntryBox Control.

Syntax *Object*.**PictureWidth** [= *NewWidth*]

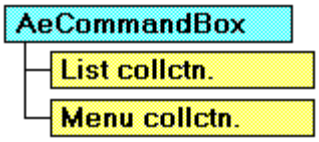
Part	Description
Object	A **AeEntryBox** Control.
NewWidth	An Integer specify the indentation width in pixels.

Remarks If the Picture property is left clear, the **PictureWidth** can still be used for indenting the text.

AeCommandBox Control

Members

- Properties
- Methods
- Events

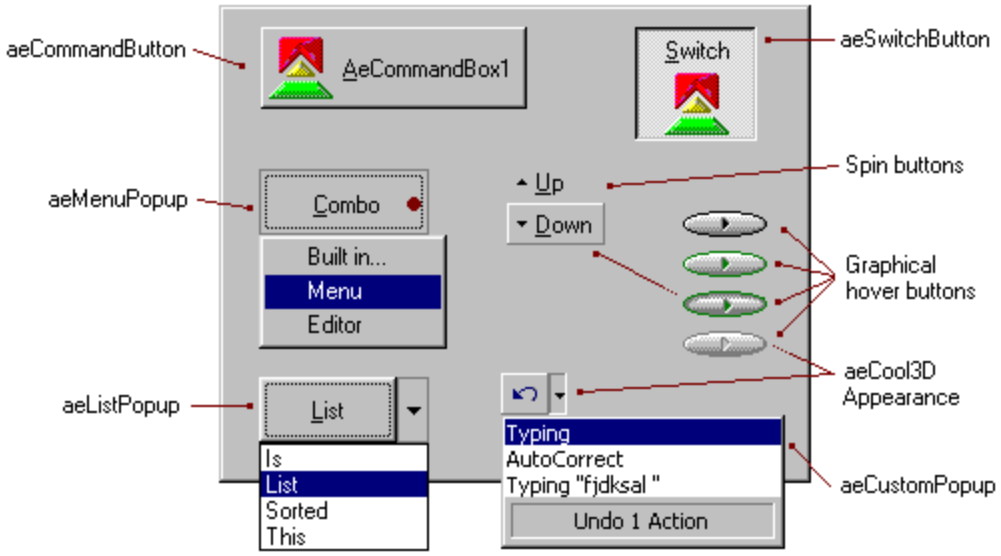


The AeCommandBox is an extended CommandButton control.

The Style property adds support for other types of button including an on/off switch style, and buttons that popup a List, Menu or your own custom popup forms.

Other enhancements to the standard button are:

+ HighlightPicture allows a web style effect when the mouse hovers over the control.
+ The HasComboButton property works with the popup button styles, and provides a second combo button for showing th
+ RepeatRate allows the user to hold the mouse down and have Click events triggered repeatedly at a specified interval,
+ TextAlignment & PictureAlignment provide complete control over the positioning of both text and graphics.
+ ShowBorder and ShowFocus for more control over the appearance.



AeCommandBox Methods

CreatePopupForm
Drag
Move
Popup
Refresh
SetFocus
ShowWhatsThis
ZOrder

AeCommandBox Properties

Appearance
BackColor
Cancel
Caption
ComboWidth
Container
Default
DisabledPicture
DragIcon
DragMode
Enabled
Font
ForeColor
HasComboButton
Height
HelpContextId
HighlightPicture
Index
Left
List
MaskColor
Menu
MouseIcon
MousePointer
Name
Object
Parent
Picture
PictureAlignment
PopupForm
PopupSymbol
PressedPicture
RepeatRate
ShowBorder
ShowFocus
Style
TabIndex
TabStop
Tag
TextAlignment
ToolTipText
Top
Value
Visible
WhatsThisHelpID
Width

AeCommandBox Events

Click
ComboClick
DragDrop
DragOver
GotFocus
KeyDown
KeyPress
KeyUp
ListClick
LostFocus
MenuClick
MenuHighlight
MenuPopup
MouseDown
MouseMove
MouseUp

ComboClick Event

Occurs when the users clicks the second combo button of a AeCommandBox control.

Syntax **Private Sub** *Object*_**ComboClick**()

Part	Description
Object	A **AeCommandBox** control.

Remarks This event is only raised if HasComboButton is set to True.

The click event for the combo button is raised before the MouseUp event, in contrast to the standard button which waits till after the mouse is released.

ListClick Event

Occurs when the user selects an item from the popup List of a AeCommandBox control.

Syntax **Private Sub** *Object*_**ListClick**(*Index* **As Integer**)

Part	Description
Object	A **AeCommandBox** Control.
Index	An Integer expression identifying the index of the list item clicked.

ComboWidth Property

Returns or sets the width of the combo button of a AeCommandBox control.

Syntax *Object*.**ComboWidth** [= *New_ComboWidth*]

Part	**Description**
Object	A valid **AeCommandBox** Control.
New_ComboWidth	A Single expression representing the width of the combo button in pixels.

Remarks Setting the **ComboWidth** to 0 (default), uses the width of the systems vertical scrollbar. The ComboWidth is ignored if HasComboButton is set to False.

Picture Property

Returns or sets the graphical image displayed on a AeCommandBox control or AeButton object.

Syntax **Set** *Object*.**Picture** [= *New_Picture*]

Part	Description
Object	A **AeCommandBox** control or **AeButton** object.
New_Picture	A valid **Picture** expression.

Remarks At design-time you can use the Properties window to load a graphic from a file. The graphics set at design-time are saved in the application.

At run-time, any valid **Picture** object can be used such as the Visual Basic **LoadPicture** function to load a graphic from file, the Picture property of another control, or a picture variable you have created.

Other optional picture properties are provided for disabled, pressed and highlighted button states. If the Picture property alone is set, the other pictures are created automatically.

Note: You can adjust the PictureAlignment and/or TextAligment properties to position the pictures and text as required.

Value Property

Returns or sets the state of a AeCommandBox control or AeButton object.

Syntax *Object*.**Value** [= *New_Value*]

Part	Description
Object	A **AeCommandBox** control or **AeButton** object.
New_Value	A AeButtonStates constant identifying the button state.

Settings The settings for **AeButtonStates** are:

Setting	Description
0 - aeUp	The normal resting state. Shows a raised border.
1 - aeDown	The pressed state after the user has pressed the mouse button down. Shows an inset border.
2 - aeToggled	The alternative resting state for switch buttons that are pressed.

Remarks The **Value** property is set according to the mouse and keyboard events raised by the user, and can be used to monitor the **AeButton**'s state.

The **Value** can also be set at run-time to manually change the **AeButton** state.

Style Property

Returns or sets the type of behaviour of a AeCommandBox control.

Syntax *Object*.**Style** [= *New_Style*]

Part	**Description**
Object	A **AeCommandBox** control.
New_Style	A AeCommandBoxStyles expression

Settings The settings for **AeCommandBoxStyles** are:

Setting	**Description**
0 - aeCommandButton	Standard CommandButton behaviour.
1 - aeSwitchButton	Button with two states, aeUp and aeToggled. Clicking the button toggles between the two states.
2 - aeListPopup	Clicking the button pops up a List.
3 - aeMenuPopup	Clicking the button pops up a Menu.
4 - aeCustomPopup	clicking the button pops up a custom form.

Remarks

RepeatRate Property

Returns or sets the rate at which a AeCommandBox control or AeButton object raises **Click** events when the mouse is held down over the control.

Syntax *Object*.**RepeatRate** [= *New_RepeatRate*]

Part	Description
Object	A **AeCommandBox** control or **AeButton** object.
New_RepeatRate	An Integer expression represent the repeat rate in milliseconds at which the Click event is raised. A value of 0 disables repeated Click events.

Remarks The **RepeatRate** is used to create 'spinning' or 'scrolling' buttons that perform an action repeatedly whilst the user holds the mouse down. The RepeatRate is set, for example, in the aeSpinBox style AeEntryBox control for spinning numeric values.

The value set is the number of milliseconds between each Click event, ie. the setting 500 will auto-repeat twice a second.

Setting the value to 0 (default) disables the auto-repeat feature.

TextAlignment, PictureAlignment Properties

Returns or sets the alignment of the text and pictures in a AeCommandBox control or AeButton object.

Syntax *Object*.**TextAlignment** [= *New_Alignment*]

Object.**PictureAlignment** [= *New_Alignment*]

Part	Description
Object	A **AeCommandBox** control or **AeButton** object.
New_Alignment	An AeButtonAlignments constant representing the type of alignment.

Settings The settings for **AeButtonAlignments** are:

Setting	Description
0 - aeAlignLeft	Aligned on the left side, centered vertically.
1 - aeAlignRight	Aligned on the right side, centered vertically.
2 - aeAlignTop	Aligned at the top, centered horizontally.
3 - aeAlignBottom	Aligned at the bottom, centered horizontally.
4 - aeAlignCenter	Default. Centered vertically and horizontally.

Remarks The **PictureAlignment** property, which determines the alignment of the **Picture**, takes precendence over the **TextAlignment** property, ie. if both picture and text are aligned to the right, the picture will be right-most.

As a convenience, the alignments are changed when the combination of **Picture** and **Caption** properties are altered. When either picture or caption is set, but not both, the alignment is set to aeAlignCenter. When both picture and caption are set, both alignments are set to aeAlignLeft.

Word-wrapping is not supported.

DisabledPicture, PressedPicture, HighlightPicture Properties

Returns or sets the custom graphic images to display in a AeCommandBox control or AeButton object, when the button is in active states.

Syntax **Set** *Object*.**DisabledPicture** [= *New_Picture*]
Set *Object*.**PressedPicture** [= *New_Picture*]
Set *Object*.**HighlightPicture** [= *New_Picture*]

Part	Description
Object	A **AeCommandBox** control or **AeButton** object.
New_Picture	A valid Picture expression

Remarks The **DisablePicture** is used when the button's **Enabled** is False. The **PressedPicture** is used when the button is pressed. The **HighlightPicture** is used in conjunction with the aeCool3D Appearance property, providing the image to use when the mouse 'hovers' over the button.

If any or all of these pictures are not set, the Picture property is used to create standard button pictures automatically. The Graphical style used by the standard CommandButton doesn't need to be set and is not included.

Tip: By setting the ShowBorder and ShowFocus properties to False, you can create highly graphical buttons using the pictures alone to illustrate the various states. Note also that the PressedPicture is moved a pixel down and to the right only when **ShowBorder** is True.

HasComboButton Property

Returns or sets whether a second combo button is added to a AeCommandBox control with a popup **Style**.

Syntax *Object*.**HasComboButton** [= *Setting*]

Part	Description
Object	A **AeCommandBox** Control.
Setting	A Boolean expression

Remarks When **HasComboButton** is False (default), and the Style property is set to one of the popup styles, the button itself triggers the popup form when clicked. When set to True, a second combo button is added to the right of the main button, which shows the popup form.

The **Click** and ComboClick events can be used to identify which button is clicked.

The PopupSymbol is used in both cases, either added to right of the main button, or as the picture of the combo button.

Popup Method

Shows the popup list, menu or form of a AeCommandBox control.

Syntax *Object*.**Popup**

Part	**Description**
Object	A **AeCommandBox** Control.

Remarks This method only applies to the popup Styles.
If the popup form is already visible, calling the **Popup** method again, hides it.

PopupSymbol Property

Returns or sets the predefined picture in a AeCommandBox control, used to illustrate a popup form.

Syntax *Object*.**PopupSymbol** [= *New_Symbol*]

Part	Description
Object	A **AeCommandBox** control.
New_Symbol	A AeSymbols constant identifying the symbol to display.

Settings The settings for **AeSymbols** are:

Setting	
0 - aeNoSymbol	
1 - aeDownSymbol	
2 - aeUpSymbol	
3 - aeLeftSymbol	
4 - aeRightSymbol	
5 - aeCalendarSymbol	
6 - aeClockSymbol	
7 - aeSearchSymbol	
8 - aeNewSymbol	
9 - aeEllipsisSymbol	
10 - aeCarriageReturnSymbol	
11 - aeHelpSymbol	
12 - aeAsteriskSymbol	
13 - aePlusSymbol	
14 - aeMinusSymbol	
15 - aeTickSymbol	
16 - aeCrossSymbol	
17 - aeFirstSymbol	
18 - aeLastSymbol	
19 - aeMaximiseSymbol	
20 - aeMinimiseSymbol	
21 - aeCloseSymbol	
22 - aeStopSymbol	
23 - aeRecordSymbol	
24 - aePlaySymbol	
25 - aePauseSymbol	
26 - aeFastForwardSymbol	
27 - aeRewindSymbol	
28 - aeLEDOffSymbol	
29 - aeLEDRedSymbol	
30 - aeLEDGreenSymbol	
31 - aeComboSymbol	

Remarks The default symbol is the aeComboSymbol.

When HasComboButton is False, the **PopupSymbol** is shown on the right of the main button. When **HasComboButton** is True, it is shown on the combo button.

ShowFocus Property

Returns or sets whether a AeCommandBox control changes its appearance when it receives the focus.

Syntax *Object*.**ShowFocus** [= *Setting*]

Part	Description
Object	A **AeCommandBox** Control.
Setting	A Boolean expression

Remarks Setting **ShowFocus** to False prevents the button from change appearance when it gets or loses the focus.

When **ShowFocus** is True (default), and the Appearance is aeFlat or aeAuto3D, a focus rectangle is drawn on the button when it has the focus. If the **Appearance** is aeCool3D, the button appears highlighted, as if the mouse was over it.

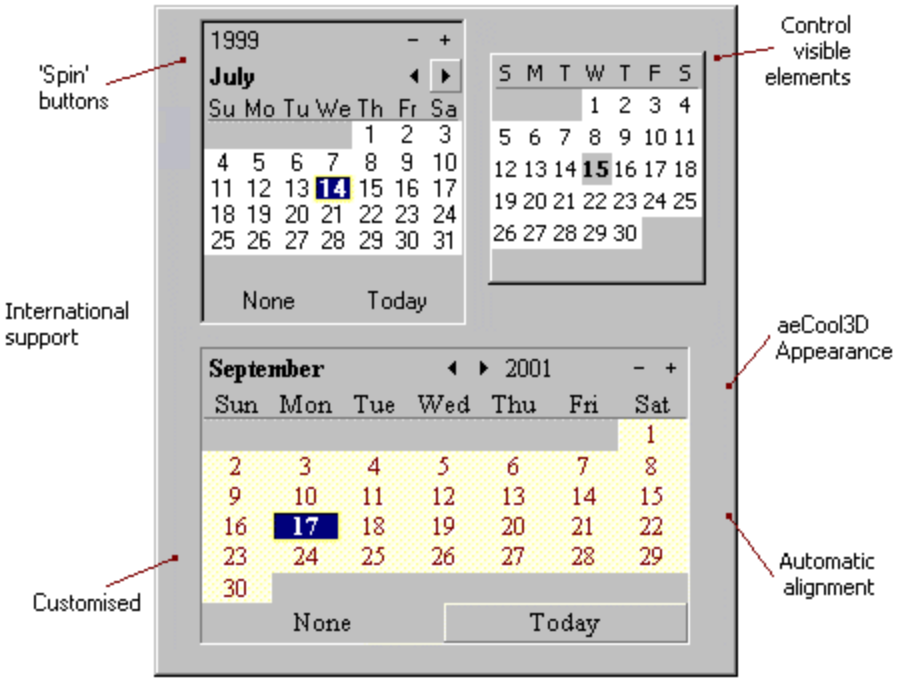
AeCalendar Control

Members

- Properties
- Methods
- Events

Features include:

- Year and Month titles can each be hidden and automatically align side by space if wide enough. Day titles also expand to 2 or 3 letters.
- International support for different system locales.
- Customised with Font and Color properties.



AeCalendar Methods

Refresh

AeCalendar Properties

Appearance
BackColor
BorderStyle
ButtonItem
DefaultDate
CurrentMonth
CurrentYear
Enabled
Font
ForeColor
Font
hWnd
MinValue
MaxValue
ShowMonth
ShowNone
ShowToday
ShowYear
Value

AeCalendar Events

Change
Click
KeyDown
KeyPress
KeyUp
MonthChange
MouseDown
MoveMove
MouseUp

Click Event

Occurs when the user selects a day in the AeCalendar control.

Syntax **Private Sub** *Object*_**Click**()

Part	Description
Object	A **AeCalendar** control.

Remarks

Change Event

Occurs whenever the selected date of an AeCalendar control changes.

Syntax **Private Sub** *Object*_**Change**()

Part	Description
Object	A **AeCalendar** control.

Remarks The Change event is not triggered when the displayed month is changed. Use the MonthChange event to handle this.

Value Property

Returns or sets the selected date of an AeCalendar control.

Syntax *Object*.**Value** [= *New_Value*]

Part	Description
Object	A **AeCalendar** control.
New_Value	A Date expression represent the selected date.

Remarks The **Value** can be used to examine the currently selected date, or set the date to a new value. The selected date is shown in bold text.

As an alternative to setting the **Value**, the DefaultDate property can be used to set a pre-defined value relative to the current date.

Note: It is possible for the displayed month to be different from the selected date, in which case none of the days will appear selected.

CurrentMonth, CurrentYear Properties

Returns or sets the currently displayed month and year of an AeCalendar control.

Syntax *Object*.**CurrentMonth** [= *New_Value*]

Object.**CurrentYear** [= *New_Value*]

Part	Description
Object	A **AeCalendar** control.
New_Value	A Integer expression from 1 to 12 for the CurrentMonth, or a valid 4 digit number for the CurrentYear.

Remarks The user can select the displayed month and year at run-time by selecting the values in the month and year titles, or pressing the PageUp/PageDown and Cursor keys in the table of days.

Setting the Value or DefaultDate properties also changes the month and year displayed.

The MonthChange event is triggered whenever the month or year changes.

DefaultDate Property

Sets the current date of an AeCalendar control to a pre-defined value relative to the current date.

Syntax *Object*.**DefaultDate** = *New_DefaultDate*

Part	Description
Object	A **AeCalendar** control.
New_DefaultDate	A AeDefaultDates constant.

Settings The settings for **AeDefaultDates** are:

Setting	Description
0 - aeToday	Selects today.
1 - aeTomorrow	Selects tomorrow.
2 - aeYesterday	Selects yesterday.
3 - ae1WeekToday	Selects 1 week from today.
4 - ae2WeeksToday	Selects 2 weeks from today.
5 - ae3WeeksToday	Selects 3 weeks from today.
8 - ae4WeeksToday	Selects 4 weeks from today.

Remarks Setting the DefaultDate to one of the above constants is equivalent to setting the Value property to the corresponding date.

The DefaultDate is write-only.

MonthChange Event

Occurs whenever the month or year displayed by a AeCalendar Control changes.

Syntax **Private Sub** *Object*_**MonthChange**()

Part	Description
Object	A **AeCalendar** Control.

ShowMonth, ShowYear Properties

Returns or sets whether the month and year titles, of an AeCalendar control, are visible.

Syntax *Object*.**ShowMonth** [= *New_Value*]
Object.**ShowYear** [= *New_Value*]

Part	Description
Object	A **AeCalendar** control.
New_Value	A Boolean expression.

Remarks When both titles are visible, the year is aligned above the month box, unless the control is wide enough to align the two titles side by side.

ShowToday, ShowNone Properties

Returns or sets the Today and/or None button commands at the bottom of a AeCalendar control are visible.

Syntax *Object*.**ShowNone** [= *Setting*]

Part	Description
Object	A **AeCalendar** Control.
Setting	A Boolean expression

Remarks

ButtonItem Property

Returns one the buttons in a AeCalendar control, as a AeButton object.

Syntax *Object*.**ButtonItem**(*Index*)

Part	Description
Object	A **AeCalendar** control.
Index	An Integer expression identifying the button as follows: 1 - Month increase button 2 - Month decrease button 3 - Year increase button 4 - Year decrease button 5 - Select None button 6 - Select Today button

Remarks

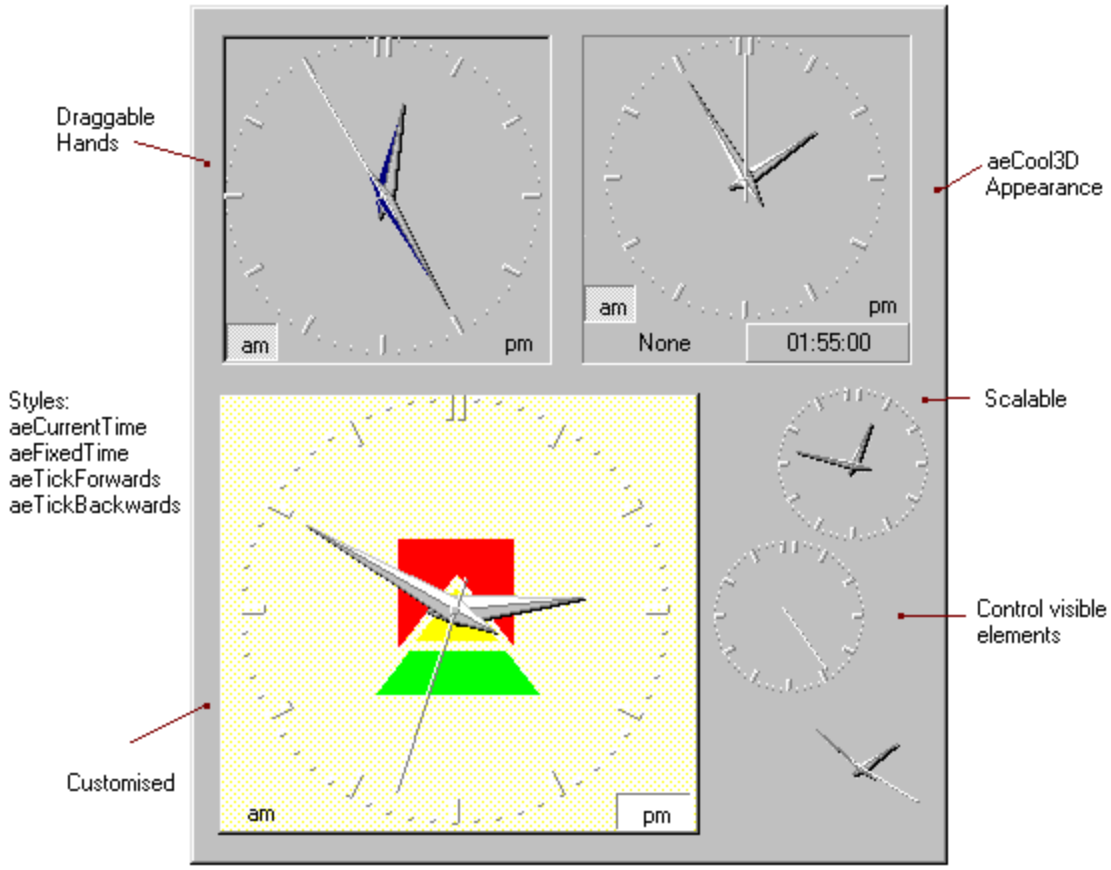
AeClock Control

Members

- Properties
- Methods
- Events

Features include:

- Each clock hand can be dragged around with the mouse. Set the time with 2 mouse clicks!
- Can be set to display the time like a clock, or tick backwards like a stopwatch.
- Each of the elements can be hidden for custom use.



AeClock Methods

ClearAlarm
HitTest
Refresh
SetAlarm

AeClock Properties

AllowEdit
Appearance
BackColor
BorderStyle
DataField
DataSource
ForeColor
Font
MaskColor
MouseIcon
MousePointer
Picture
ShowAmPm
ShowFrame
ShowHourHand
ShowMinuteHand
ShowNone
ShowSecondHand
ShowTime
Style
UseMaskColor
Value

AeClock Events

Alarm
Click
Change
DblClick
KeyDown
KeyPress
KeyUp
MouseDown
MouseMove
MouseUp

Change Event

Occurs whenever the time displayed by an AeClock control changes.

Syntax **Private Sub** *Object*_**Change**()

Part	Description
Object	An **AeClock** control.

Remarks

Value Property

Returns or sets the time dispalyed by an AeClock control.

Syntax *Object*.**Value** [= *New_Value*]

Part	**Description**
Object	An **AeClock** control.
New_Value	A Date expression representing the time to be displayed.

Remarks Times are stored with the **Date** data type. These variables use a double precision number to store both Date and Time by using the number day number starting from 30/12/1899 plus a fraction from 0 to 1 corresponding to the time of day.

Style Property

Returns or sets the type of time to display in a AeClock Control.

Syntax *Object*.**Style** [= *New_Style*]

Part	Description
Object	An **AeClock** Control.
New_Style	A AeClockStyles constant representing the style.

Settings The settings for **AeClockStyles** are:

Setting	Description
0 - aeFixedTime	The time displayed remains fixed.
1 - aeCurrentTime	The current time is displayed and updated every second.
2 - aeTickForwards	The clock ticks forwards every second, but may be set to any time.
3 - aeTickBackwards	The clock ticks backwards every second, but may be set to any time.

Remarks For controls requiring the user to enter a time, the aeFixedTime is generally used. Setting the Style to aeCurrentTime, makes the control behave like a standard clock. The aeTickForwards and aeTickBackwards styles are useful for other customised clock behaviours.

The Style setting effects the behaviour of the AllowEdit property in determining how the user can change the time.

The Style and AllowEdit properties can be changed at run-time, for example creating stop-watch controls.

Alarm Event

Occurs when a previously set alarm is triggered by a AeClock Control.

Syntax **Private Sub** *Object*_**Alarm**(*Name* **As String**, *Message* **As String)**

Part	Description
Object	An **AeClock** Control.
Name	The name of the alarm.
Message	An optional message associated with the alarm.

Remarks The SetAlarm method is used to set an alarm. The event is triggered whenever the time displayed passes the alarm time.

ClearAlarm Method

Clears an alarm which has previously set for an AeClock Control.

Syntax *Object*.**ClearAlarm** *Name*

Part	**Description**
Object	A **AeClock** Control.
Name	A String specifying the name of the alarm to be cleared.

Remarks The SetAlarm method is used to set an alarm.

SetAlarm Method

Sets an alarm to occur at a specified time for a AeClock Control.

Syntax *Object*.**SetAlarm** *Name*, *AlarmTime*, *Message*

Part	Description
Object	A **AeClock** Control.
Name	A unique String for identifying the alarm.
AlarmTime	A Date expression specifying the time the alarm is triggered.
Message	Optional. A String specifying a message to be associated with the alarm.

Remarks When the time displayed passes the specified AlarmTime, the Alarm event is triggered. If AllowEdit is set to True, then the user can also trigger an alarm by changing the time displayed.

Once the alarm is triggered, it is removed. The ClearAlarm method can also be used to cancel an alarm before it is triggered.

HitTest Method

Returns the clock hand at the specified co-ordinates of a AeClock Control. Returns a AeClockHands constant.

Syntax *variable* = *Object*.**HitTest**(*X*, *Y)*

Part	Description
Object	A **AeClock** Control.
X	A Long value identifying the X co-ordinate in pixels.
Y	A Long value identifying the Y co-ordinate in pixels.

Settings The settings for **AeClockHands** are:

Setting	Description
0 - aeNoHand	None of the hands are at the position.
1 - aeHourHand	The Hour hand is at the position.
2 - aeMinuteHand	The Minute hand is at the position.
3 - aeSecondHand	The Second hand is at the position.

Remarks Since the clock hands may be overlapping, the HitTest methods examines the aeMinuteHand first, followed by the aeMinuteHand, then the aeSecondHand.

This method is used internally for dragging the hands with the mouse, and so the regions tested for each hand may be larger than their graphical representations on the screen.

ShowAmPm Property

Returns or sets whether the AM and PM buttons of a AeClock Control are displayed.

Syntax *Object*.**ShowAmPm** [= *Setting*]

Part	Description
Object	A **AeClock** Control.
Setting	A Boolean expression.

Remarks When the buttons are not visible, there is no way for the user to determine whether the time is AM or PM. An alternative view of the time should therefore be given in, for example, a Label control.

ShowFrame Property

Returns or sets whether the minute and hour tick marks of a AeClock Control are displayed.

Syntax *Object*.**ShowFrame** [= *Setting*]

Part	**Description**
Object	A **AeClock** Control.
Setting	A Boolean expression

ShowHourHand Property

Returns or sets whether the hour hand of a AeClock Control is displayed.

Syntax *Object*.**ShowHourHand** [= *Setting*]

Part	**Description**
Object	A **AeClock** Control.
Setting	A Boolean expression

ShowMinuteHand Property

Returns or sets whether the minute hand of a AeClock Control is displayed.

Syntax *Object*.**ShowMinuteHand** [= *Setting*]

Part	**Description**
Object	A **AeClock** Control.
Setting	A Boolean expression

ShowSecondHand Property

Returns or sets whether the second hand of a AeClock Control is displayed.

Syntax *Object*.**ShowSecondHand** [= *Setting*]

Part	**Description**
Object	A **AeClock** Control.
Setting	A Boolean expression

AllowEdit Property

Returns or sets whether the user can change the time displayed in a AeClock Control.

Syntax *Object*.**AllowEdit** [= *Setting*]

Part	Description
Object	A **AeClock** Control.
Setting	A Boolean expression

Remarks When AllowEdit is set to True, the user can change the time displayed by using the mouse to drag individual hands around, or using the keyboard with the Left/Right and Plus/Minus keys to rotate the minute hand, or Up/Down and PageUp/PageDown keys to rotate the hour hand.

Clicking the left mouse button will move the minute hand to that position (unless clicked on another hand). Clicking the right mouse button will move the hour hand. This provides a quick way to enter times with two mouse clicks.

If the Style property is set to aeCurrentTime, then AllowEdit will usually need to be set to False, since the time will always revert to the current time. The aeTickForwards and aeTickBackwards styles work with AllowEdit set to True, and will continue to tick from the new time set by the user.

ShowTime, ShowNone Properties

Returns or sets the selected time and/or None button commands at the bottom of a AeClock control are visible.

Syntax *Object*.**ShowNone** [= *Setting*]

Part	**Description**
Object	A **AeCalendar** Control.
Setting	A Boolean expression

Picture Property

Returns or sets the image to display on the background of a AeClock Control.

Syntax **Set** *Object*.**Picture** [= *New_Picture*]

Part	**Description**
Object	A **AeClock** Control.
New_Picture	A valid Picture expression. Windows bitmaps and icons supported.

Remarks Use the MaskColor property to define the transparent areas.

AeButton Object, AeButtons Collection

Members

Properties
Methods

Containers

AeEntryBox control
AeCalendar control

The **Buttons** collection contains the strip of buttons on the right of an **AeEntryBox** control.

Syntax *Control*.**Buttons**

Control.**Buttons**(*Index*)

Part	**Description**
Control	A container control.
Index	An Integer expression referencing the **Button**'s index in the collection, starting at 1.

Remarks At design-time, buttons can be added and configured using the custom property pages. Right click the control, and select Properties to view the custom property pages. At run-time, the **Buttons** collection of a control can be refreferenced in code to change the button properties.

Example: The following example loops through each button in an AeEntryBox named aebAuthor:

```
Dim oButton As AeButton
Dim n As Integer
For n = 1 To ebxAuthor.Buttons.Count
    Set oButton = aebAuthor.Buttons(n)
    MsgBox "Button " & CStr(n) & "'s caption is " _
            & oButton.Caption
Next n
```

Note: At run-time, after changing the **Buttons** colleciton, the **Refresh** method may need to be called to update the control.

AeButton Object, AeButtons Collection Properties

Properties Methods Overview >> Legend

- Actions
- Appearance
- BackColor
- DefaultWidth
- Caption
- Count
- CustomWidths
- DisabledPicture
- Enabled
- Height
- HighlightPicture
- Index
- Item
- Left
- MaskColor
- Picture
- PictureAlignment
- PressedPicture
- RepeatRate
- ShowBorder
- Style
- Symbols
- TextAlignment
- ToolTipText
- Top
- Value
- Width

AeButton Object, AeButtons Collection Methods

- Add
- Clear
- HitTest
- Refresh
- Remove

- ○ Object only
- ○ Collection only
- ○ Object and Collection

HitTest Method

Returns the index of the AeButton located at the specified coordinates.

Syntax *Variable* = *Object*.**HitTest**(*X*, *Y)*

Part	Description
Object	A **AeButtons** collection.
X	A Single expression, representing the horizontal position from the left.
Y	A Single expression, representing the vertical position from the top.

Remarks This method is useful when examining mouse movement or implementing drag-and-drop operations.

Add Method

Adds a AeButton object to a **AeButtons** collection and returns a reference to the newly created **AeButton** object.

Syntax **Set** *Button* = *Object*.**Add**([*Position*])

Part	Description
Button	A **AeButton** variable.
Object	A **AeButtons** collection.
Position	Optional. An Integer expression specifying the position of the new button.

Remarks You can add **AeButton**'s at design-time by using the custom property pages of the control. At run-time, you could use the following sample code:

```
Dim NewButton as AeButton
Set NewButton = EntryBox1.Add()
```

If the Position part is not specified, the new **AeButton** is added to the end of the collection.

Remove Method

Removes a specified AeButton object from a **AeButtons** collection.

Syntax *Object*.**Remove** *Index*

Part	**Description**
Object	A **AeButtons** collection.
Index	An Integer expression specifying the **Index** of the **AeButton** to be removed.

Remarks To remove all the buttons, use the Clear method.

Clear Method

Clears all the AeButton objects in a **AeButtons** collection.

Syntax *Object*.**Clear**

Part	**Description**
Object	A **AeButtons** collection.

Remarks To remove a specific button, use the Remove method.

Item Method

Returns a reference to a specified AeButton object in a **AeButtons** collection.

Syntax *Object*.**Item**(*Index*)
Object(*Index*)

Part	Description
Object	A **AeButtons** collection.
Index	An Integer specifying the **Index** of the button.

DefaultWidth Property

Sets the width of AeButton objects within a **AeButtons** collection.

Syntax *Object*.**DefaultWidth** = *Value*

Part	Description
Object	A **AeButtons** collection.
Value	A Single expression specifying the width in Pixels, of the buttons.

Remarks New buttons added to a collection are given a default **Width** equal to the **DefaultWidth** property.

By default, the **Width** is set to the width of a vertical scroll-bar. You can change this width at design-time in the property page, or at run-time from code.

AeButton objects can be given different widths. Note however that when the **DefaultWidth** is changed, all the existing buttons in the collection are resized.

Count Property

Returns the number of AeButton objects in a **AeButtons** collection.

Syntax *Object*.**Count**

Part	**Description**
Object	A **AeButtons** collection.

Symbols Property

Returns or sets a pre-defined images to display in a AeButton object or AeCommandBox control.

Syntax *Object*.**Symbols**(*Index*) [= *Icon*]

Part	Description
Object	A **AeButtons** collection.
Index	An Integer expression, representing the index of the button within it's collection.
Icon	An AeSymbols constant, defining which image to display.

Settings The settings for **AeSymbols** are:

Setting	
0 - aeNoSymbol	
1 - aeDownSymbol	
2 - aeUpSymbol	
3 - aeLeftSymbol	
4 - aeRightSymbol	
5 - aeCalendarSymbol	
6 - aeClockSymbol	
7 - aeSearchSymbol	
8 - aeNewSymbol	
9 - aeEllipsisSymbol	
10 - aeCarriageReturnSymbol	
11 - aeHelpSymbol	
12 - aeAsteriskSymbol	
13 - aePlusSymbol	
14 - aeMinusSymbol	
15 - aeTickSymbol	
16 - aeCrossSymbol	
17 - aeFirstSymbol	
18 - aeLastSymbol	
19 - aeMaximiseSymbol	
20 - aeMinimiseSymbol	
21 - aeCloseSymbol	
22 - aeStopSymbol	
23 - aeRecordSymbol	
24 - aePlaySymbol	
25 - aePauseSymbol	
26 - aeFastForwardSymbol	
27 - aeRewindSymbol	
28 - aeLEDOffSymbol	
29 - aeLEDRedSymbol	
30 - aeLEDGreenSymbol	
31 - aeComboSymbol	

Remarks Setting the **Symbol** property is equivalent to setting the **Picture** property.
Since all the icons are small, the system resources used are minimal.

Actions Property

Returns or sets the action to be performed when a AeButton object is clicked.

Syntax *Object*.**Actions** [= *Index*]

Part	Description
Object	A valid **AeButton** Object.
Index	An Integer expression
Action	An AeEntryBoxActions constant identifying the action.

Settings The settings for **AeEntryBoxActions** are:

Setting	Description
0 - aeCustomAction	No action is performed.
1 - aeIncreaseValue	The value for numeric data types is increased by value of the Step property.
2 - aeDecreaseValue	As aeIncrement, but decreases the value.
3 - aePopupList	Shows the popup list.
4 - aePopupCalendar	Shows the popup calendar.
5 - aePopupClock	Shows the popup clock.
6 - aePopupMenu	Shows the popup menu.
7 - aePopupCustom	Shows the custom popup form.
8 - aeSetTrue	Sets the value to True.
9 - aeSetFalse	Sets the value to False.

Remarks

Style Property

Returns or sets the type of a AeButton object.

Syntax *Object*.**Style** [= *Value*]

Part	Description
Object	A **AeButton** object.
Value	A AeButtonStyles constant.

Settings The settings for **AeButtonStyles** are:

Setting	Description
0 - aePushButton	Standard button
1 - aeToggleButton	Button toggles between aeUp and aeToggled states.
2 - aeGroupButton	As aeToggleButton, but only one group button can be selected at a time.

Remarks The aeGroupButton **Style** is only applicable to buttons that are part of a collection. Any neighbouring buttons in the collection with the aeGroupButton style are treated as part of the group.

CustomWidths Property

Returns or sets the width of a button within a AeButtons collection.

Syntax *Object*.**CustomWidths**(*Index*) = *Setting*

Part	Description
Object	A **AeButtons** collection.
Index	An Integer identifying which button in collection, starting at 1.
Setting	A Boolean expression.

Remarks Setting the **CustomWidths** to 0 (default), uses the default width for all buttons stored in the DefaultWidth property.

AeList Collection

Members	**Containers**
Properties	AeEntryBox control
Methods	AeCommandBox control

The **AeList** collection maintains a list of text items associated with a control and a ListBox for displaying the items. Searching and formatting features are also provided.

Syntax *Control*.**List**

Control.**List**(*Index)*

Part	**Description**
Control	A **AeEntryBox** or **AeCommandBox** control.
Index	An Integer expression referencing the index of the item in the list, starting at 1.

Remarks The **List** property returns an AeList collection with its own propertys and methods. This differs from a standard ComboBox control. The equivalent members are listed below:

ComboBox control	**AeList collection**
AddItem	Add
RemoveItem	Remove
List(Index)	Item(Index)
ListIndex	Index
ListCount	Count

Note: The popup ListBox control is not created until the first time it is displayed. Therefore, items can be added and utilised without consuming many system resources. A common example of this is creating a aeComboBox style **AeEntryBox** with some list items added, and changing the button to do something other than show the popup list. The AutoComplete feature will help the user type in common values by searching the list, whilst the combo button is used to show a more advanced search form for example.

AeList Methods

Add
Clear
FindFirst
Remove

AeList Properties

Count
Data
Index
Item
ItemData
NoVisibleItems
Sorted
Width

Add Method

Adds an item to a AeList collection. Returns the Index of the new item.

Syntax *Object*.**Add** *Text*[, *Index*]

New_Index = *Object*.**Add**(*Text*[, *Index*[, *Data*]])

Part	**Description**
New_Index	An Integer variable.
Object	A **AeList** object.
Text	A String expression specifying the item contents.
Index	Optional. An Integer expression, specifying the position to put the new item in.
Data	Optional. An Integer expression, specifying the ItemData value of the new item.

Remarks If successful, the method returns the Index of the new item, or 0 if the new item could not be added.

If the **AeList** has the Sorted property set to True, the Index part is ignored, and the new item is positioned in the sorted order.

Remove Method

Removes an item from a AeList collection.

Syntax *Object*.**Remove** *Index*

Part	Description
Object	A **AeList** object.
Index	An Integer expression specifying the index of the item to be removed, starting with 1.

Remarks To remove all the items in a AeList, use the Clear method.

Clear Method

Removes all the items in a AeList collection.

Syntax *Object*.**Clear**

Part	Description
Object	A **AeList** Object.

FindFirst Method

Searches the items in a AeList collection for a string. Returns the index of the first item found.

Syntax *Index* = *Object*.**FindFirst**(*Value*[[, *Start*], *ExactMatch*])

Part	Description
Index	An Integer variable.
Object	A **AeList** object.
Value	A String expression specifying the text to search for.
Start	Optional. An integer expression specifying the position in the list to start searching from. If ommitted, the search begins from the beginning.
ExactMatch	Optional. A boolean expression specifiying whether an exact match is returned. The default is False.

Remarks The FindFirst operation searches the items for an exact match if it can find one, or the first item that starts with the same characters. If no item is found, the method returns 0. Specifying the ExactMatch parameter, forces the search to find an exact match only.

The text case is ignored.

Count Property

Returns the number of items in a AeList collection.

Syntax *Variable* = *Object*.**Count**

Part	**Description**
Object	A **AeList** object.

Item Property

Returns or sets the text of the specified item in a AeList collection.

Syntax *Object*(*Index*) [= *Text*]

Object.**Item**(*Index*) [= *Text*]

Part	Description
Object	A **AeList** object.
Index	An Integer expression specifying the item's index, starting with 1.
Text	A String expression representing the text of the specified item.

ItemData Property

Returns or sets a numeric data value for the items in a AeList object.

Syntax *Object*.**ItemData**(*Index*) [= *Value*]

Part	Description
Object	A **AeList** object.
Index	An Integer expression specifying the item's index, starting with 1.
Value	A Long expression representing the data value of the specified item.

Remarks The **ItemData** is useful for associating a numeric value with each item in the list, for example, to cross-reference with a database table.

NoVisibleItems Property

Returns or sets the **NoVisibleItems** that a AeList collection will display when the ListBox pops up.

Syntax *Object*.**NoVisibleItems** [= *Value*]

Part	Description
Object	A **AeList** object.
Value	An Integer expression

Remarks The standard VB ComboBox only displays up to 8 items in the drop-down list. The **NoVisibleItems** property allows you to specify how many items are displayed. For example, a drop-down list of months can have 12 items.

Setting the **NoVisibleItems** to 0 (default) will show however many items there are in the list, up to the default maximum 12.

Index Property

Returns or sets the **i**ndex of the currently selected item in a AeList object.

Syntax *Object*.**Index** [= *Value*]

Part	**Description**
Object	A **AeList** object.
Value	A Long expression representing the data value of the specified item.

Remarks The **AeList** object retains the currently selected item, even if the list is not visible, and continues to remain synchronised with the edit area in the **AeEntryBox** control. This allows quick validation that the text entered by the user exists in the list. For example…

```
Private Sub AeEntryBox1_Change()

    If AeEntryBox1.List.Index = 0 Then
        'Text entered not in list
    End If

End Sub
```

Sorted Property

Returns or sets whether the items in a AeList object are sorted alphabetically.

Syntax *Object*.**Sorted** [= *New_Sorted*]

Part	Description
Object	A **AeList** object.
New_Sorted	A Boolean expression

Remarks The **Sorted** property can be changed at design-time and run-time. The sort is not case-sensitive

Width Property

Returns or sets the width of the popup List in a AeList object.

Syntax *Object*.**Width** [= *New_Width*]

Part	Description
Object	A **AeList** object.
New_Width	An Integer expression, specifiying the width in pixels. A value of 0 (default), uses the width of the control.

Data Property

Returns the Data of the currently selected item in a AeList object.

Syntax ItemData = *Object*.**Data**

Part	**Description**
Object	A **AeList** object.
ItemData	An integer variable.

Aebacus Popup Control Library

The Aebacus Popup Control Library is an ActiveX control file (.OCX), containing 4 controls:

	AeEntryBox Control	Cool Input control combining ComboBox, MaskedEdit, DatePicker, SpinBox, and MS Forms functionality, and adding data validation, flat-look, auto-complete, popup clocks, custom popup forms, menus, and configurable combo buttons.
	AeCommandBox Control	An advanced CommandButton control. Includes flexible alignment, 'hover' effects, popup forms, built in combo button, plus…
	AeCalendar Control	User-friendly calendar control for selecting dates. Includes spin buttons, extended keyboard handling, and formatting features.
	AeClock Control	Flexible clock control allows the user to enter a time with the clock hands. Also displays the current time or ticks backwards for stopwatches.

+ All controls provide a aeCool3D Appearance for building modern **'flat-look'** user interfaces (like Office 97 and Internet Explorer)
+ Developer-friendly **Property Pages** are provided for quick and easy configuration, including an improved menu editor.
+ Online **documentation**, context-sensitive help, standard naming conventions, and loads of helpful samples.
+ Dependent only the Visual Basic run-time library.
+ Extensible at run-time through a comprehensive **object model** and informative events.
+ Ideal for building **Data entry forms** 'on-the-fly' using control arrays, centralized event handling, and built in data type conversion.
+ **Data bound** Value property for quick connection to databases.
+ **ActiveX container** support, for embedding the controls in a variety of environments, including Office 97, Internet Explorer 3/4, Visual Studio, Outlook etc.

User Guide:

Installation	Installation instructions and file descriptions.
Getting Started	Summary of the 4 controls for getting up and running quickly.
Creating Custom Popup Forms	Details about creating your own custom popup forms with examples.
Creating Menus at Run-Time	Example code for creating menus at run-time, and handling menu events.
ActiveX Container Support	Issues relating to using the controls in ActiveX containers other than VB.

Support:

Registration & Support	How to register, and support information.
System Requirements	Platforms supported and design environments.
Dependencies & Distribution	List of file dependencies and distribution details.
Terms & Conditions	Software Product License.

Appearance Property

Returns or sets the **Appearance** of an AeEntryBox, AeCommandBox, AeCalendar or AeClock control, and an AeButton object.

Syntax *Object*.**Appearance** [= *Value*]

Part	Description
Object	A valid control.
Value	An **AeAppearances** constant:

Settings The settings for **AeAppearances** are:

Setting	Description
aeFlat	Flat border styles.
aeAuto3D	Standard 3D appearance.
aeCool3D	Flat appearance when inactive, becoming 3D when it becomes the active control or the mouse moves over it.

Remarks The standard **Appearance** property had been enhanced throughout the Aebacus Controls with a third aeCool3D setting in addition to the standard aeFlat and aeAuto3D settings.

The aeCool3D style is an active appearance that "comes alive" when the mouse moves over the control or it receives the focus. This appearance provides a modern look-and-feel similar to Microsoft Office 97 and Internet Explorer 4.

MaskColor Property

Returns or sets the colour to use for the transparent part of images drawn on the AeEntryBox, AeCommandBox, AeCalendar or AeClock controls.

Syntax *Object*.**MaskColor** [= *Color*]

Part	Description
Object	A **Button** object.
Color	A RGB value representing the mask color.

Remarks Colors can be set using the color palette in the property pages, or using the **RGB** and **QBColor** functions in code.

The default **MaskColor** is vbGrey, RGB(192,192,192).

ShowBorder Property

Returns or sets whether the border of a AeCommandBox is visible.

Syntax *Object*.**ShowBorder** [= *Setting*]

Part	Description
Object	A valid control.
Setting	A Boolean expression

BorderStyle Property

Returns or sets the type of border surrounding the AeEntryBox, , AeCalendar and AeClock controls.

Syntax *Object*.**BorderStyle** [= *NewStyle*]

Part	**Description**
Object	A valid control.
NewStyle	A **AeBorderStyles** constant

Settings The settings for **AeBorderStyles** are:

Setting	**Description**
0 - aeNoBorder	The control border hidden.
1 - aeInset	A 3D inset border
2 - aeRaised	A 3D raised border

Remarks The border is also effected by the Appearance setting

Registration & Support

For the most up to date information on registration and support, please visit our web site, or send an enquiry by email, at:

Web Address: http://www.aebacus.demon.co.uk/

E-Mail Address: enquiries@aebacus.demon.co.uk

Technical support is only available to registered owners.

Registered owners receive:

+ Single user design-time license
+ Unlimited run-time license (no royalties)
+ Free Delivery via the internet
+ Free Updates
+ Discounted Upgrades
+ Technical Support

Installation

The evaluation copy is distributed in a self-extracting file (AeDemo.exe). Keep this file in case you need to re-install the program.

To install, run the AeDemo.exe file, which extracts the set up files into a temporary directory and starts the installation. After completing the installation, you can delete the set-up files from the temporary directory.

The program can be uninstalled using the Windows 'Add/Remove Programs' control panel, or using the shortcut added to the Start menu.

The following files are installed into the directory specified during the setup:

Filename	**Description**
AePopup(D).ocx	ActiveX control library.
AePopup.hlp/.cnt/.gid	This help file.
Popup Properties.exe/.vbp	Sample program for all 4 controls (source code in sub-dir).
Entry Outlook.exe/.vbp	AeEntryBox sample program (source code in sub-dir).
Value Bound.exe/.vbp	Data bound version of Entry Outlook (requires DAO v3.51).
Global Clock.exe/.vbp	AeClock sample program (source code in sub-dir).
Popup Browser.htm	Web page sample with embedded controls.
Data.mdb/.txt	Databases for Entry Outlook samples.
TimeZones.txt	Database for Global Clock sample.
Readme.txt	Overview and maintenance history.
Vendor.txt	Information for software distributors.
UnInst.isu	Information required for uninstall.

For your safety, no files are installed in the Windows or System folders.

System Requirements

Platform Support

The library runs on 32-bit Windows platforms, specifically:

- Windows 95
- Windows 98
- Windows NT 3.51
- Windows NT 4
- Windows NT 2000

Design Environments

The controls may also used in any application which supports ActiveX controls. The following ActiveX containers have been tested:

- Visual Basic 5 & 6
- Office 97 VBE
- Access 97
- Word 97
- Excel 97
- Outlook 97/98
- Internet Explorer 4/5
- ActiveX Control Pad (Web Pages)

This number of design environments that support ActiveX controls is continuing to grow. See the ActiveX Container Support page for more details.

Dependencies & Distribution

Dependencies

The Aebacus Popup Controls is only dependant on the Visual Basic 5 run-time library. This is available with Visual Basic 5 (SP2/3), or as a separate installable file from the Microsoft web site at http://support.microsoft.com/download/support/mslfiles/Msvbvm50.exe. The VB5 run-time is also pre-installed with the Windows 98 operating system.

Msvbvm50.exe installs the following files, which ship with Visual Basic 5.0 Service Pack 2 and Service Pack 3, into the Windows System folder:

File	Version
Msvbvm50.dll	5.00.4319
Oleaut32.dll	2.20.4118
Olepro32.dll	5.0.4118
Stdole2.tlb	2.20.4118
Asycfilt.dll	2.20.4118
Comcat.dll	4.71

Distribution

The registered version provides an unlimited run-time license, which enables you to include the controls in your applications, and distribute those applications freely without having to pay royalty fees.

If distributing the controls in a VB5 application, then only the AePopup.ocx file needs to be included, since the VB5 run-time will already be included by the installation software (e.g. Application Setup Wizard).

If distributing the controls in a VB6 application, or any other ActiveX container, then the installation software will also need to install the VB5 run-time. This can be done by either including the above files in the installation routine, or simply including the Msvbvm50.exe with the distribution set, and running it prior to starting the installation.

The AePopup.dep file (registered version only), can be used by installation packages, to automatically determine the file dependencies.

The controls are marked safe for initialisation and scripting (registered version only), and so may be distributed and installed through Web browsers in Intranet applications.

Getting Started

For the best overview, it is recommended you play with the “Popup Properties.exe” sample, and the custom Property Pages at design-time.

AeEntryBox

The AeEntryBox provides the same standard properties, methods and events as the Visual Basic TextBox and ComboBox controls. The main enhancements to these controls are:

+ The Style property is provided for a quick way to configure the control’s properties for common types of data entry (*aeComboBox* for list pickers, *aeDateBox* for date pickers etc.). The property is slightly unusual, in that it automatically configures other properties. Subsequently changing those effected properties, sets the Style to *aeCustomBox*.
+ The standard ComboBox’s popup list has been implemented as a List collection object for simpler coding. Also provides a FindFirst searching method, a custom Width property, and better co-operation with text area.
+ In addition to the list, there’s also a popup calendar, clock, menu, and even a custom popup form. Each button is assigned an Action to perform when clicked, e.g. *aePopupList* to show the list or *aePopupCalendar* to show the calendar.
+ A number of extra properties are available for data entry applications. The DataType is used in conjunction with the Value property for built in conversion, the FormatString exposes VB’s formatting capabilities, and Min/MaxValue, and IsValid properties are for validation.
+ Setting AutoComplete to True (default), not only completes matching entries from the List, but also completes dates and times.
+ The Buttons collection is completely configurable, providing functionality similar to a whole toolbar! The buttons’ Custom Property Page best illustrates this.

AeCommandBox

The AeCommandBox provides the same standard properties, methods and events, as the Visual Basic CommandButton control. The main enhancements include:

+ In addition to the default *aeCommandButton*, the Style property includes *aeSwitchButton*, *aeListPopup*, *aeMenuPopup*, and *aeCustomPopup* styles.
+ Setting the HasComboButton property to True, adds a second combo button to the right of the main button, like those in the IE and Office toolbars.
+ The TextAlignment and PictureAlignment properties should satisfy most alignment needs.
+ Setting the ShowFocus property to False, prevents the focus rectangle from appearing on the button.
+ The HighlightPicture, has been added to the standard Picture properties, for mouse ‘hover’ effects.

The AeCalendar and AeClock controls, are also provided as standalone controls. These are suited more specifically to date/time applications.

Creating Custom Popup Forms

The AeEntryBox and AeCommandBox controls, both provide a mechanism for attaching your own customised popup forms. Custom forms are attached to the controls at run-time through the CreatePopupForm method. This provides a powerful way to enhance the functionality of the controls with limitless possibilities.

Note: *Modeless* forms are the easiest and most flexible type of custom popup form, but should be approached with CAUTION. Some containers do not support showing modeless forms. *Modal* forms are far safer and supported by all containers, but are limited to childless windows (i.e. they can't have any contained controls).

Modeless Popup Forms

An example best illustrates using the **CreatePopupForm** method. A main form contains a single **AeEntryBox** control, with the aeCustomPopup Action assigned to the combo button. A second form containing two standard buttons, labelled OK and Cancel, is used for the popup form.

```
'Main Form
'=========
Private Sub Form_Load()
   Set frmPopup.PopupForm = AeEntryBox1.CreatePopupForm(frmPopup.hWnd)
End Sub

'Custom Popup Form (frmPopup)
'============================
Public PopupForm As AePopupForm

'OK button
Private Sub Command1_Click()
   PopupForm.Value = "OK clicked"
   PopupForm.Hide
End Sub

'Cancel button
Private Sub Command2_Click()
   PopupForm.Cancelled = True
   PopupForm.Hide
End Sub
```

Your custom popup form must communicate with the control, to tell it when a value has been selected. This is done through the **AePopupForm** object, returned by the **CreatePopupForm** method.

Clicking the combo button will display the frmPopup window below the AeEntryBox control. Pressing OK results in the text changing to "OK clicked", pressing Cancel will leave the text unchanged.

The **AeEntryBox** control shows the form either when a button, with the aePopupCustom Actions assigned, is clicked, or when the Action method is called from code. The **AeCommandBox** control shows the form when the Style is set to aeCustomPopup, or when the Popup method is called from code.

If the popup form contains child windows (i.e. like the two CommandButtons in this example), then it can be activated, and is considered *Modeless*.

Note: Some containers may not support modeless forms (the VB property pages and Internet Explorer are examples of this).

Modal Popup Forms

If the popup form does not contain any child windows (i.e. a form containing only windowless controls like Labels, or a single control like a ListBox), the it can not be activated, and is considered *Modal*.

Although modal forms are more limited in terms of functionality, they are better suited for custom popup forms. Since they don't receive the focus, the user interface is improved, and also, modal forms are supported by VB property pages and Internet Explorer.

The following example illustrates a modal custom popup form, using a ListBox control instead of a second form used in the modeless example. The ListBox control is placed on the same form as the AeEntryBox control, and its Visible property is set to False.

```
'Main Form
'=========
Private m_PopupForm As AePopupForm

Private Sub Form_Load()
   Set m_PopupForm = AeEntryBox1.CreatePopupForm(ListBox1.hWnd)
End Sub

Private Sub ListBox1_Click()
   PopupForm.Value = ListBox1.List(ListBox1.ListIndex)
   PopupForm.Hide
End Sub
```

Note: Whilst the standard controls can be used as modal popup forms, more complicated controls, such as the ListView and TreeView common controls, may not support the CreatePopupForm method.

Creating Menus at Run-Time

Both the AeEntryBox and AeCommandBox controls, provide a Menu object that can be built at design-time or run-time.

Using the standard VB menu, a 'seed' menu item needs to be created at design-time, and used to create extra menu items at run-time. The AeMenu does not impose this restriction, and allows new menus, and submenus, to be created at run-time.

The following example illustrates this. Create a new project, and place a AeCommandBox and a Label control on the form. Change the AeCommandBox.Style property to aePopupMenu. The code creates the menu when the form loads, and changes the Label Caption in response to the menu items being selected AND highlighted.

```
Private Sub Form_Load()

    'construct a new menu
    With AeCommandBox1.Menu

        'standard collection methods used to add items
        .Add "&Built"
        .Add "&at"
        .Add "&Runtime"
        .Add "", aeSeperatorMenuItem

        'create a submenu, and keep reference for adding items to
        Dim oMenu As AeMenuItem
        Set oMenu = .Add("&Submenu", aeSubMenuItem, "Submenu1")

        'define some Radio and Check menu items, with Keys defined
        oMenu.Submenu.Add "Option &1", aeRadioMenuItem, "Option1"
        oMenu.Submenu.Add "Option &2", aeRadioMenuItem, "Option2"
        oMenu.Submenu.Add "", aeSeperatorMenuItem
        oMenu.Submenu.Add "&Check", aeCheckMenuItem, "Check"

    End With

End Sub

Private Sub AeCommandBox1_MenuClick(MenuItem As AePopup.AeMenuItem)

    'use the item's Parent and Index to identify
    If MenuItem.Parent Is AeCommandBox1.Menu Then
        Select Case MenuItem.Index
            Case 1:  Label1 = "Build selected"
            Case 2:  Label1 = "at selected"
            Case 3:  Label1 = "Runtime selected"
        End Select
    Else
        Select Case MenuItem.Index
            Case 1:  Label1 = "Option1 selected"
            Case 2:  Label1 = "Option2 selected"
            Case 4
                If MenuItem.Checked Then
                    Label1 = "Check selected"
                Else
                    Label1 = "Check cleared"
                End If
        End Select
    End If

End Sub

Private Sub AeCommandBox1_MenuHighlight(MenuItem As AePopup.AeMenuItem)

    'or, use the item's Key to identifiy
```

```
    Select Case MenuItem.Key
        Case "Option1":   Label1 = "Use the first option"
        Case "Option2":   Label1 = "Use the second option"
        Case "Check":   Label1 = "Toggle the check option"
    End Select

End Sub
```

ActiveX Container Support

The controls have been tested on a wide variety of ActiveX containers, and we are continuing to ensure compatibility with the growing number of development environments that support ActiveX controls.

This page contains a number of issues related to ActiveX containers, other than Visual Basic.

Extender Name Conflicts

Most design environments provide a "wrapper" control around every ActiveX control. When you access a property or method of the control, the wrapper passes this on to the actual control behind the scenes. A problem arises when both the wrapper and the ActiveX control provide a property with the same name, since the compiler has no way of knowing which property to use, and defaults to the wrapper's property. To get around this, the wrapper provides an Object property, that forces the compiler to use the Active control's property.

A known example of this is with MS Access, which provides a Value property for any ActiveX control. This conflicts with the AeEntryBox control, which provides it's own Value property, so the following code…

```
Debug.Print AeEntryBox1.Value
```

…prints the Value stored by the Access 97 wrapper (which may produce an error if not used). To get to the actual AeEntryBox1.Value property, instead use…

```
Debug.Print AeEntryBox1.Object.Value
```

AeEntryBox Property Pages

The General, Behaviour, and Buttons tabs on the AeEntryBox custom Property Pages are inter-dependent, for example, the DataType property on the General tab, effect a number of properties on the Behaviour tab.

Under Visual Basic, making changes in one tab, will correctly notify other tabs to refresh themselves. However most other ActiveX containers, notably MS Office, do NOT refresh the other tabs.

This can lead to some problems with the AeEntryBox. In particular, the Style option on the General should be changed to aeCustomBox whenever changes are made in the Buttons tab, and this can lead to conflicts if the Style is left unchanged. To work around this, always change the Style to aeCustomBox manually, BEFORE making any changes in the Buttons tab.

The properties on the Behaviour tab, can be refreshed by closing and re-opening the property pages, whenever the Style or DataType is changed on the General tab.

Storing Binary Properties in HTML Web Pages

When employing ActiveX controls in web pages, the properties are stored in <PARAM> tags, for example the following HTML code displays a default AeCommandBox with it's Caption set to "Run"…

```
<OBJECT CLASSID="clsid:139767BC-DE13-11D2-AFDE-B1B99C86281E"
        ID="AeCommandBox1"
        WIDTH=100
        HEIGHT=30>
        <PARAM NAME="Caption" VALUE="Run">
</OBJECT>
```

Since the properties are stored as a String, it is not possible to store binary values, such as the Picture and Font properties. There is a convenient workaround for the Font, since by default, the controls use the Font of the container, which in web browsers, is determined by the <FONT> and other formatting

tags.

The Picture properties require the use of tool capable of persisting binary properties in HTML tags, such as the *ActiveX Control Pad* which is freely available from Microsoft's web site. This provides a simple designer for building ActiveX controls using the built in property pages (including the Picture tab). When a Picture property is stored, the program will create a DATA attribute for the <OBJECT> tag like "DATA:application/x-oleobject;BASE64,xxx…" which stores all the properties in binary format (gobbledegook).

Terms & Conditions

AEBACUS POPUP CONTROLS
version 1.02
19 March, 1999

Software Product License

The Aebacus Popup Controls is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The Aebacus Popup Controls is licensed, not sold.

LICENSE AGREEMENT. The registered Aebacus Popup Controls includes a design time license, permitting a single user to use the control in a development environment, and an unlimited run-time license, permitting free distribution of the Aebacus Popup Controls ActiveX control file with developed applications. No royalties are charged. Distribution of the registered Aebacus Popup Controls installation file, or license keys, are forbidden by law. A separate multi-user license is required for persons, other than the registered owner, who wish to use the Aebacus Popup Controls in a design-time environment.

COPYRIGHT. All title and copyrights in and to the Aebacus Popup Controls (including but not limited to any images, photographs, animations, video, audio, music, text, source code, and applets, incorporated into the Aebacus Popup Controls), the accompanying printed materials, and any copies of the Aebacus Popup Controls, are owned by Bayley Consultancy Ltd.. The Aebacus Popup Controls is protected by copyright laws and international treaty provisions.

Disclaimer Agreement

Users of Aebacus Popup Controls must accept this disclaimer of warranty:

"Aebacus Popup Controls is supplied as is. The author disclaims all warranties, expressed or implied, including, without limitation, the warranties of merchantability and of fitness for any purpose. The author assumes no liability for damages, direct or consequential, which may result from the use of Aebacus Popup Controls."

Aebacus Popup Controls is a "shareware program" and is provided at no charge to the user for evaluation. Feel free to share it with your friends, but please do not give it away altered or as part of another system. The essence of "user-supported" software is to provide personal computer users with quality software without high prices, and yet to provide incentive for programmers to continue to develop new products. If you find this program useful and find that you are using Aebacus Popup Controls and continue to use Aebacus Popup Controls after a reasonable trial period, you must make a registration to Bayley Consultancy Ltd.. The registration fee will license one copy for use on any one computer at any one time. You must treat this software just like a book. An example is that this software may be used by any number of people and may be freely moved from one computer location to another, so long as there is no possibility of it being used at one location while it's being used at another. Just as a book cannot be read by two different persons at the same time.

Commercial users of Aebacus Popup Controls must register and pay for their copies of Aebacus Popup Controls within 30 days of first use or their license is withdrawn. Site-License arrangements may be made by contacting Bayley Consultancy Ltd..

Anyone distributing Aebacus Popup Controls for any kind of remuneration must first contact Bayley Consultancy Ltd. for authorisation. This authorisation will be automatically granted to distributors recognised by the (ASP) as adhering to its guidelines for shareware distributors, and such distributors may begin offering Aebacus Popup Controls immediately (However Bayley Consultancy Ltd. must still be advised so that the distributor can be kept up-to-date with the latest version of Aebacus Popup Controls.).

You are encouraged to pass a copy of Aebacus Popup Controls along to your friends for evaluation. Please encourage them to register their copy if they find that they can use it. All registered users will receive a copy of the latest version of the Aebacus Popup Controls.

AePopupForm Object

Members	Containers
Properties	AeEntryBox control
Methods	AeCommandBox control

The **AePopupForm** object is used internally to handle all popup forms in the Aebacus Popup Control library, including custom popup forms from your own application that you can add to the controls using the CreatePopupForm method.

Both the **AeEntryBox** and **AeCommandBox** controls have the the **CreatePopupForm** method which returns a **AePopupForm** object. The returned object can then be used in your application to notify the control that a value in your custom form was selected or that the form was cancelled.

See the the Creating Custom Popup Forms topic for an example and more details.

Remarks Advanced Windows messaging techniques are used to provide the popup forms with a modern display mechanism similar to that used in Microsoft's Outlook 98. The standard ComboBox captures the mouse whenever the popup list is displayed, with the result that clicking another control in the application to do something else, merely hides the popup form and 'eats' the mouse click. The **AePopupForm** does not capture the mouse. You can even move the parent window around the screen whilst the popup form is displayed.

AePopupForm Methods

Hide

AePopupForm Properties

Cancelled
Modal
MoveWithParent
RightAlign
Value
Visible

AePopupForm Events

Change
Load
Unload

Hide Method

Hides a window previously shown by a AePopupForm object.

Syntax *Object*.**Hide**

Part	Description
Object	A **AePopupForm** Object.

Modal Property

Returns or sets whether the form of a AePopupForm object is displayed modally by disabling the parent desktop window whilst displayed.

Syntax *Object*.**Modal** [= *Setting*]

Part	Description
Object	A **AePopupForm** Object.
Setting	A Boolean expression

Remarks This method is NOT completely implemented.

The property is provided as a safety guard against future problems showing modeless forms in ActiveX containers. The **Modal** property is currently only applicable to popup forms containing child windows (i.e. those that can't be prevented from receiving the focus).

When **Modal** is True, the desktop window, that the popup form is be displayed over, is disabled during the period the form is displayed. This is equivalent to using the VB Form's Show method with the vbModal argument.

Visible Property

Returns whether the form of a AePopupForm object is currently displayed.

Syntax *Variable* = *Object*.**Visible**

Part	**Description**
Variable	A Boolean variable
Object	A **AePopupForm** Object.

Value Property

Returns or sets a value that the user selected from a AePopupForm object.

Syntax *Object*.**Value** [= *NewValue*]

Part	**Description**
Object	A **AePopupForm** Object.
NewValue	A Variant expression

Remarks This property is used to notify parent controls of the value selected from custom popup forms in your application. When the popup form closes, the AeEntryBox will set it's own **Value** to that of the AePopupForm unless it's Cancelled property has been set to True.

Cancelled Property

Returns or sets whether a displayed from from a AePopupForm object was hidden without the user selecting a value.

Syntax *Object*.**Cancelled** [= *Setting*]

Part	Description
Object	A **AePopupForm** object.
Setting	A Boolean expression

Remarks The **Cancelled** property is reset to False whenever the popup form is displayed. If the user cancels a custom popup form in your application, the **Cancelled** property should be set to True before calling the Hide method.

RightAlign Property

Returns or sets whether a displayed from from a AePopupForm object aligns to the bottom-right corner of the parent control.

Syntax *Object*.**RightAlign** [= *Setting*]

Part	Description
Object	A **AePopupForm** object.
Setting	A Boolean expression.

Remarks By default, popup forms are aligned to the bottom-left of the parent control.

Change Event

Occurs whenever the Value of a AePopupForm object changes.

Syntax **Private Sub** *Object*_**Change**()

Part	Description
Object	A **AePopupForm** object.

Unload Event

Occurs when a displayed form from a AePopupForm object is about to be hidden.

Syntax **Private Sub** *Object*_**Unload**(*Cancel* **As Boolean**,)

Part	Description
Object	A **AePopupForm** object.
Cancel	A Boolean variable set to False. Setting Cancel to True will prevent the popup form from hiding.

Load Event

Occurs when the form of a AePopupForm object is about to be displayed.

Syntax **Private Sub** *Object*_**Load**(*Modal* **As Boolean**)

Part	Description
Object	A **AePopupForm** Object.
Modal	A Boolean expression signifying whether the form is being loaded modally.

Remarks The Modal argument is used internally only, and can be ignored for the moment.

MoveWithParent Property

Returns or sets whether the window shown by a AePopupForm object is moved whenever the parent desktop window is moved.

Syntax *Object*.**MoveWithParent** [= *Setting*]

Part	Description
Object	A **AePopupForm** Object.
Setting	A Boolean expression

Remarks Set this property to False, will hide the popup form whenever the mouse is clicked, similar in behaviour to the popup forms in Outlook 98. Note that the mouse is still not 'captured' if MoveWithParent is set to False.

If you find the popup form is being left behin, when the parent form is moved, setting MoveWithParent to True, will prevent it. This can happed in MDI child forms, for example.

AeMenu Object

Members
- Properties
- Methods

Containers
- AeEntryBox control
- AeCommandBox control

The **AeMenu** object provides an enhanced standard Menu control. The object maintains a collection of AeMenuItem objects.

Syntax *Control*.**Menu**

Part	Description
Control	A **AeEntryBox** or **AeCommandBox** control.

Remarks The **AeMenu** can be set up at desgn-time using the Menu property page, or at run-time using the object model.

The nested model for **AeMenu** and **AeMenuItem** objects provide easier handling of menu items and submenus. A **AeMenuItem** can contain a nested **AeMenu** object for building submenus, reflecting the actual menu structure. Also, submenus can be created at run-time and menu items can be moved around.

The following example illustrates the code required to access the 2nd menu item of a root menu's 2nd item's submenu's 1st item's submenu.

```
Dim oMenuItem As AeMenuItem
Set oMenuItem = AeCommandBox1.Menu(2).Submenu(1).Submenu(2)
```

The Highlight event is provided for notification of when the mouse highlights a menu item.

AeMenu Methods

Add
Clear
MoveItem
Remove

AeMenu Properties

Count
hMenu
Item
Parent
RightAlign

AeMenu Events

Click
Highlight
Show

Highlight Event

Occurs when menu item belonging to a AeMenu object becomes highlighted.

Syntax **Private Sub** *Object*_**Highlight**(*MenuItem* **As AeMenuItem**)

Part	Description
Object	A **AeMenu** object.
MenuItem	The **AeMenuItem** object that became highlighted.

Click Event

Occurs when menu item belonging to a AeMenu object is selected by the user.

Syntax **Private Sub** *Object*_**Click**(*MenuItem* **As AeMenuItem**,)

Part	Description
Object	A **AeMenu** Object.
MenuItem	The **AeMenuItem** object that was selected.

Show Event

Occurs whenever a AeMenu object pops up up.

Syntax **Private Sub** *Object*_**Show**(*Menu* **As AeMenu**)

Part	**Description**
Object	A **AeMenu** Object.
Menu	The **AeMenu** that popped up.

Remarks The Show event is triggered for the root menu and submenus alike.

Add Method

Adds a new menu item to a AeMenu object. Returns a reference to the new AeMenuItem object.

Syntax **Set** *MenuItem* = *Object*.**Add**(*Caption*[, *ItemType*[, *Key*[, *Position*]]])

Part	Description
MenuItem	A **AeMenuItem** variable.
Object	A **AeMenu** object.
Caption	The text to display on the menu item.
ItemType	Optional. A AeMenuItemTypes constant specifying the type of menu item to add. The default is aeDefaultMenuItem.
Key	Optional. A unique String expression identifying the menu item.
Position	Optional. The position of the new menu item.

Remarks

Remove Method

Removes a menu item from a AeMenu object.

Syntax *Object*.**Remove** *Index*

Part	Description
Object	A **AeMenu** object.
Index	A String or Integer expression identifying the **Key** or **Index**, respectively, of the menu item to remove.

Clear Method

Removes all menu item from a AeMenu object.

Syntax *Object*.**Clear**

Part	Description
Object	A **AeMenu** object.

MoveItem Method

Moves a menu item to another position in a AeMenu object.

Syntax *Object*.**MoveItem** *Index*, *Position*,

Part	**Description**
Object	A **AeMenu** Object.
Index	A String or Integer expression identifying the **Key** or **Index**, respectively, of the item to be moved.
Position	An Integer expression specifying the position to move to.

Item Property

Returns a AeMenuItem object belonging to a AeMenu object.

Syntax **Set** *MenuItem* = *Object*.**Item**(*Index*)

Set *MenuItem* = *Object*(*Index*)

Part	**Description**
MenuItem	A **AeMenuItem** variable.
Object	A **AeMenu** Object.
Index	A String or Integer expression identifying the **Key** or **Index**, respectively, of the item.

Count Property

Returns the number of menu items in a AeMenu object.

Syntax *Variable* = *Object*.**Count**

Part	**Description**
Variable	An Integer variable
Object	A **AeMenu** object.

hMenu Property

Returns the Windows menu handle of a AeMenu object.

Syntax *Variable* = *Object*.**hMenu**

Part	**Description**
Variable	An Integer variable
Object	A **AeMenu** object.

Remarks This property is used in calls to the Windows API.

Parent Property

Returns the parent AeMenuItem of submenu AeMenu objects, or Nothing if the menu is the root menu.

Syntax *Object*.**Parent**

Part	**Description**
Object	A **AeMenu** object.

Remarks The **Parent** property is useful identifying the position in complex menu structures, of the **AeMenu** object passed in the menu events.

RightAlign Property

Returns or sets whether the root menu of a AeMenu object aligns to the bottom-right corner of the parent control.

Syntax *Object*.**RightAlign** [= *Setting*]

Part	Description
Object	A **AeMenu** Object.
Setting	A Boolean expression, which if False (default), aligns the menu down the left side of the parent control.

Remarks

AeMenuItem Object

Members
Properties

Containers
AeMenu

A **AeMenuItem** object is created for each menu item in a **AeMenu** object.

Syntax *Object*.**Item**(*Index*)

Object(*Index*)

Part	**Description**
Object	A **AeMenu** object.
Index	A String or Integer expression identifying the **Key** or **Index**, respectively, of the item.

Remarks The aeCheckMenuItem and aeRadioMenuItem Styles provide automatic handling of check marks and radio group marks.

Items with the aeSubMenuItem **Style** provide a Submenu object for natural coding of the menu structure within the object model.

AeMenuItem Properties

Properties Overview >>

Caption
Checked
Enabled
HelpContextId
Index
Style
Key
Parent
Submenu
Visible

Key Property

Returns or sets a unique string for identifying a AeMenuItem object.

Syntax *Object*.**Key** [= *Name*]

Part	**Description**
Object	A **AeMenuItem** object.
Name	A unique String expression.

Remarks The Key need only be unique within the item's parent **AeMenu** object. Other menus in the structure can have items with the same key.

Style Property

Returns or sets the type of a AeMenuItem object.

Syntax *Object*.**Style** [= *New_Type*]

Part	Description
Object	A **AeMenuItem** object.
New_Type	A AeMenuItemStyles constant identifying the type.

Settings The settings for **AeMenuItemStyles** are:

Setting	Description
0 - aeDefaultMenuItem	Normal menu item.
1 - aeCheckMenuItem	Includes a check mark on the left which toggles on/off when selected.
2 - aeRadioMenuItem	Includes a radio mark on the left which toggles on when selected, and off when a neighbouring aeRadioMenuItem is selected.
3 - aeSeperatorMenuItem	An inactive item displaying a separator bar.
4 - aeSubMenuItem	An item that includes a submenu **AeMenu** object.

Remarks The first four types correspond with those found in the standard Menu control.

Submenu Property

Returns a AeMenu object belonging to a aeSubMenuItem style AeMenuItem object.

Syntax *Object*.**Submenu**

Part	**Description**
Object	A **AeMenuItem** Object.

Remarks The **Submenu** property is only available for items with the aeSubMenuItem Style.

The following example prints whether the second item on a submenu is selected.

```
Debug.Print AeCommandBox1.Menu(1).Submenu(2).Checked
```

Parent Property

Returns the parent AeMenu object of a AeMenuItem object.

Syntax *Object*.**Parent**

Part	**Description**
Object	A **AeMenuItem** Object.

Remarks The Parent property can be used to identify the position of a menu item within the menu structure. For example, if you have two Submenus with similar items, then the code to handle the MenuClick event might be:

```
Private Sub AeCommandBox1_MenuClick(MenuItem As AeMenuItem)

Dim ParentItem As AeMenuItem

'get reference to the submenu item using the parent's parent.
Set ParentItem = MenuItem.Parent.Parent

'if no Parent, then item is on the root menu
If ParentItem Is Nothing Then

Else
        'use the Key to identify which submenu to use
        If ParentItem.Key = "SubmenuA" Then
                DoSomething MenuItem.Index
        Elseif ParentItem.Key = "SubmenuB" Then
                DoSomethingElse MenuItem.Index
        End If
        End If

End Sub
```