The Unicus TextBox Control

from Unicus Data Systems

Introduction

The TextBox is probably the most used control in Visual Basic. Virtually every application has at least a few of them. Unfortunately, the TextBox control that comes with Visual Basic does not provide very much built-in functionality.

You have to manually add a caption, and make sure the caption is properly positioned and aligned, and that the tab index of the textbox is one greater than the caption. If you want to move the textbox somewhere else on your form, you have to manually move the caption, and then change the tab index of both controls.

Further, if you want any validation to ensure that the user doesn't enter bad data, or formatting to display it in a user-friendly way, you have to write, test, and debug your own code to do what you need. Then you have to duplicate that effort for every single textbox on your form.

All of this adds significantly to the time it takes to develop a form. The Unicus TextBox control solves all of these problems, and more. It brings almost all the features and functionality of the Microsoft Access textbox to Visual Basic. The first thing you'll notice when you drop a Unicus TextBox control on your form is the attached caption. This alone can save a lot of time because it means you don't have to create a separate caption control for each textbox and make sure that it is properly aligned and positioned relative to the textbox. The caption supports hotkeys, and if you assign it a hotkey, it will automatically set focus to the textbox when the user activates the hotkey. The textbox will also automatically receive focus if the user clicks on the label.

The Unicus TextBox exposes a number of properties that give you, the developer, a high degree of control over its behaviour and appearance. You can position the caption to the left, right, top, or bottom of the textbox, or even hide the caption altogether. You can set the font and appearance properties of the caption separately from the textbox. You can make the control automatically select the entire contents of the textbox when it receives the focus, just like in Access. If you need precise control over the caption position, you can use the horizontal and vertical offset properties to place it exactly where you want.

The Unicus TextBox works in bound or unbound mode, just like the intrinsic TextBox control. If you use it in bound mode, you can rely on the built-in DataFormat property of the DataBindings collection. Or, in either bound or unbound mode, you can use the custom DataType, DisplayFormat, CustomFormat, DataPrecision, DataScale, and NumberOfDecimals properties to control what type of data is entered, and how it is displayed to the user.

Another very useful feature is the Value property. The Text property of the intrinsic TextBox control always returns the contents of the textbox as they are displayed. There is no way to convert that to the actual value without writing your own code. This is one of its most serious limitations. The Text property of the Unicus TextBox works exactly the same. However, the Unicus TextBox also exposes a Value property. This is the default property, and it always returns the underlying data value of the contents of the textbox, based on the DataType, regardless of how it is formatted. So you can display a percent field as 15.34%, which makes sense to the user, but the value property will return 0.1534, so you don't have to do any conversion at all before writing it to the database or using it in your code. Further, if the textbox is empty, the Text property returns an empty string, but the Value property returns Null. As a result, where normally you would have to write code similar to the following to properly handle saving a percent field to the database

```
Dim strValue As String

If IsNull(Text1.Text) Then
   rs!Field = Null
Else
   strValue = Text1.Text
   strValue = Replace(strValue, "%", "")
   If IsNumeric(strValue) Then
      strValue = strValue / 100
   Else
      strValue = 0
   End If
   rs!PercentField = strValue
End If
```

You only have to write one simple line of code

```
rs! PercentField = udsTextBox1
```

Because the Unicus TextBox does all the validation automatically, you don't have to do any of it. Finally, the Unicus TextBox exposes Before and After Update events, so you have complete control over the data that goes into and out of the control. The Value property is not updated until the AfterUpdate event, so if you don't like what the user entered, you cancel the BeforeUpdate event, and the value will not change.

Properties

Appearance

Value	Description
0	Flat
1	3D (default)

Returns or sets whether or not the textbox portion of the control is painted at run time with 3-D effects.

AutoSelectAll

Value	Description
True	Contents are automatically selected when the control receives the focus
False	Contents are not automatically selected when the control receives the focus

Boolean value that returns or sets whether the entire contents of the control are selected when the control receives the focus.

BackColor

Returns or sets the background color of the textbox portion of the control.

BorderStyle

Value	Description
0	None
1	Fixed Single

Returns or sets the border style of the textbox portion of the control.

Caption

Returns or sets the text of the caption portion of the control.

CaptionAlignHoriz

Value	Description
0	Left
1	Right
2	Center

Returns or sets the horizontal alignment of the caption. This property is affected by the CaptionPosition and TextBoxFixedWidth properties. If CaptionPosition is set to 2-Top or 3-Bottom, CaptionAlignHoriz will have the expected effect. If CaptionPosition is set to 0-Left or 1-Right, CaptionAlignHoriz will have almost no effect unless TextBoxFixedWidth is set to a non-zero value.

CaptionAlignVert

Value	Description
0	Top
1	Bottom

Returns or sets the vertical alignment of the caption. This property is affected by the CaptionPosition property. If CaptionPosition is set to 2-Top or 3-Bottom, CaptionAlignVert has no effect. If CaptionPosition is set to 0-Left or 1-Right, CaptionAlignVert will vertically align the caption to the top or bottom of the textbox.

CaptionAppearance

Value	Description
0	Flat
1	3D

Returns or sets whether or not the caption portion of the control is painted at run time with 3-D effects.

CaptionBackColor

Returns or sets the background color of the caption portion of the control.

CaptionBorderStyle

Value	Description
0	None
1	Fixed Single

Returns or sets the border style of the caption portion of the control.

CaptionFont

Returns or sets a Font object that determines the font characteristics of the caption portion of the control

CaptionFontBold

Boolean value that returns or sets whether or not the font of the caption portion of the control is bolded.

CaptionItalic

Boolean value that returns or sets whether or not the font of the caption portion of the control is italicized.

CaptionFontSize

Returns or sets the font size of the caption portion of the control.

CaptionFontStrikethru

Boolean value that returns or sets whether or not the font of the caption portion of the control has the strikethrough characteristic.

CaptionFontUnderline

Boolean value that returns or sets whether or not the font of the caption portion of the control is underlined.

CaptionForeColor

Returns or sets the forecolor of the caption portion of the control.

CaptionOffsetHoriz

Returns or sets a Long Integer value that determines the number of scale mode units that the caption portion of the control is adjusted horizontally from its original position. This property gives you complete control over how close to the textbox the caption will be positioned. Setting CaptionOffsetHoriz to a negative value will move the caption to the left. Setting it to a positive value will move it to the right.

CaptionOffsetVert

Returns or sets a Long Integer value that determines the number of scale mode units that the caption portion of the control is adjusted vertically from its original position. This property gives you complete control over how close to the top or bottom of the textbox the caption will be positioned. Setting CaptionOffsetVert to a negative value will move the caption up. Setting it to a positive value will move it down.

CaptionPosition

Value	Description
0	Positions the caption to the left of the textbox
1	Positions the caption to the right of the textbox
2	Positions the caption above the textbox

3 Positions the caption below the textbox

Returns or sets a value that determines the position of the caption portion of the control relative to the textbox portion of the control. You can position the caption on any side of the textbox, which allows you to use the control anywhere on your forms.

CustomFormat

String. If DisplayFormat is set to 12-Custom, you can use the CustomFormat property to determine how the text should be formatted for display. Whatever you set the CustomFormat property will be passed into Visual Basic's built-in Format function. If Visual Basic is able to interpret the data against the custom format, it will set the format accordingly. If the data in the textbox cannot be interpreted based on the custom format, it will simply be left as it is entered.

If DataType is set to 1-Boolean, then CustomFormat is interpreted as TruePart|FalsePart|NullPart. For example, if you want to display the phrase "Yes" if the value is true, and "No" if the value is false, you would set DataType to 1-Boolean, and CustomFormat to "Yes|No". If the value can be interpreted as a boolean, the textbox will display the text you specify. If the value cannot be interpreted as a Boolean, the textbox will interpret it as False. If you leave out the NullPart portion of the custom format then it will display a null value as Null.

DataPrecision

DataPrecision is used only if DataType is set to 5-Decimal. If DataPrecision is set to a non-zero value, then the control will not allow more than DataPrecision digits in total to be entered. If more than DataPrecision digits are entered, the control will not lose focus, and an error message will be displayed to the user. You can prevent the message from appearing, or replace it with your own message, by trapping the ControlError event and setting the SuppressMessage parameter to True. If DataType is set to anything other than 5-Decimal, DataPrecision and DataScale are ignored.

DataScale

DataScale is used only if DataType is set to 5-Decimal, and DataPrecision is set to a non-zero value. When the user tries to update the control, it will not allow more than DataPrecision digits in total to be entered. Further, it will now allow more than DataScale of those digits to be after the decimal point. DataPrecision and DataScale in udsTextBox are similar but not identical to the corresponding properties in SQL Server. If you define a field in SQL Server as Decimal, with Precision = 5 and Scale =2, then SQL Server allows no more than 3 digits to the left of the decimal, and 2 digits to the right.

If you set udsTextBox to the same properties, it will allow no more than 5 digits in total. All 5 can be to the left of the decimal and none to the right, or 4 to the left and 1 to the right, or 3 to the left and 2 to the right, but no more than 5 in total and no more than 2 to the right. If you don't care which side of the decimal the digits are entered, you can set the DataPrecision and DataScale properties to the same value.

If more than DataScale decimals are entered, the control will not lose focus, and an error message will be displayed to the user. You can prevent the message from appearing, or replace it with your own message, by trapping the ControlError event and setting the

SuppressMessage parameter to True. If DataType is set to anything other than 5-Decimal, DataPrecision and DataScale are ignored.

DataType

Value	Description
0	String. Any entry is allowed.
1	Boolean. Only data that can be interpreted as Boolean will be allowed.
2	Long Integer. Only valid long integer (SQL int) data is allowed.
3	Integer. Only valid integer (SQL smallint) data is allowed.
4	Byte. Only valid byte data (SQL tinyint) is allowed
5	Decimal. Only numeric data is allowed. Number of digits allowed is controlled by DataPrecision and DataScale.
6	Date. Only data that can be interpreted as a date is allowed.

The DataType property is used to validate what the user enters. This property eliminates the need to put any validation code in your application. You simply set this property, combined with the DataPrecision and DataScale properties if DataType is 5-Decimal, and the control does all the validation for you. If the data entered by the user cannot be interpreted against the data type you specify, the control will not lose focus, and an error message will be displayed to the user. You can prevent the message from appearing, or replace it with your own message, by trapping the ControlError event and setting the SuppressMessage parameter to True.

DisplayFormat

Value	Description	Example
0	None.	No formatting is performed.
1	Standard. Depends on NumberOfDecimals.	1,234.56
2	Fixed. Depends on NumberOfDecimals.	1234.56
3	Long Date. Depends on regional settings.	Tuesday, August 12, 2003
4	Medium Date YY.	12 Aug 03
5	Medium Date YYYY.	12 Aug 2003
6	Short Date. Depends on regional settings.	8/12/2003
7	Long Time. Depends on regional settings.	10:34:31 PM
8	Medium Time. Depends on regional settings.	10:34 PM
9	Short Time. Depends on regional settings.	22:35
10	Currency. Depends on regional settings and on NumberOfDecimals.	$59.98
11	Percent. Depends on regional settings and on NumberOfDecimals.	19.45%

12	Custom	Uses the CustomFormat property

The DisplayFormat property affects how data is displayed in the control after it loses focus. You can set it to one of the built-in formats, or set it to 12-Custom and use the CustomFormat property to define your own property. The DisplayFormat property is used in conjunction with the DataType property to determine how the Value property should be returned. If you set DisplayFormat to a value other than 0-None, the control will display the value in the appropriate format, but the value property will return the underlying data value in the data type specified by DataType. For example, if you set the DataType to 6-Date and the DisplayFormat to 3-Long Date, the Value property will return the value as data type Date, but the Text property (the displayed text) will return the underlying value formatted to Long Date as specified in the user’s Regional Settings.

Font

Returns or sets a Font object that determines the font characteristics of the textbox portion of the control

FontBold

Returns or sets whether or not the font of the textbox portion of the control is bolded.

FontItalic

Returns or sets whether or not the font of the textbox portion of the control is italicized.

FontSize

Returns or sets the font size of the textbox portion of the control.

FontStrikethru

Returns or sets whether or not the font of the textbox portion of the control has the strikethrough characteristic.

FontUnderline

Returns or sets whether or not the font of the textbox portion of the control is underlined.

ForeColor

Returns or sets the forecolor of the textbox portion of the control.

HasDataChanged

HasDataChanged returns a Boolean value that tells you whether the underlying data value in the textbox has changed since it was last updated. Because the control is automatically updated only when it loses focus or when the user presses the Enter key (if Multiline = False), it will not be updated if the user clicks directly on your data control before doing anything else. Therefore, you will need to use this property in conjunction with the UpdateValue method in the WillMove event of the ADO data control or anywhere in your code that you will cause the recordset to change records. Putting the following code in this event will cause the control to properly update the database:

```
   If udsTextBox1.HasDataChanged Then
```

```
    udsTextBox1.UpdateValue
End If
```

Height

Returns or sets the height of the control. If the CaptionPosition property is set to 0-Left or 1-Right, the textbox portion of the control will automatically expand vertically so that its height equals the Height property. If the CaptionPosition property is set to 2-Top or 3-Bottom, the textbox portion of the control will automatically expand vertically so that its height equals the Height property minus the height of the caption portion of the control.

Left

Returns or sets the distance between the internal left edge of the control and the left edge of its container.

Locked

Determines whether the textbox portion of the control can be edited by the user at runtime.

MaxLength

Returns or sets the maximum number of characters that can be entered at runtime in the textbox portion of the control.

Multiline

Returns or sets a value that determines whether the textbox portion of the control can accept multiple lines of text. If Multiline is True, the control automatically displays a vertical scrollbar, and the Enter key causes the cursor to move to the next line in the textbox portion of the control. If Multiline is False, the scrollbar disappears, and the Enter key causes the BeforeUpdate event to fire, followed by the AfterUpdate event, unless you cancel the BeforeUpdate event. When Multiline is True, the control is not updated until it loses focus. Unlike the built-in VB textbox, you can change the multiline property of the textbox at design time or run time.

NumberOfDecimals

The NumberOfDecimals property is used in conjunction with the DisplayFormat property to determine the number of decimals to display for decimal data. If DisplayFormat is set to 1-Standard, 2-Fixed, 10-Currency, or 11-Percent, you can use the NumberOfDecimals property to override the default setting of 2. If you set NumberOfDecimals greater than the number of non-zero decimals that are actually stored in the value, the control will add zeros on to the end. If you set NumberOfDecimals less than the number of non-zero decimals that are actually stored in the value, the control will use the built-in VB Round function to trim the value to the desired number of decimals.

Text

The Text property always returns the contents of the textbox portion of the control as they appear to the user. It is updated on the Change event of the textbox.

TextBoxAlignment

Returns or sets the horizontal alignment of the text in the textbox portion of the control.

TextBoxFixedWidth

Normally the textbox portion of the control automatically resized itself horizontally to fill in the entire difference between the caption width and the overall control width. You can override this behaviour, though, by setting TextBoxFixedWidth to a non-zero value. This will cause the textbox to remain the width you specify, no matter how wide or narrow you make the control. This can be useful if you have a column of controls arrayed vertically and you want all the captions to be aligned with each other at the left of the form, and all the textboxes to be the same width aligned with each other.

TextBoxWidth

TextBoxWidth returns the actual width of the textbox portion of the control at any time. If TextBoxFixedWidth is a non-zero value, then TextBoxWidth will always equal TextBoxFixedWidth. Otherwise, TextBoxWidth will vary depending on the width of the caption portion of the control and the overall width of the control. TextBoxWidth is a read-only property available at design time and run time.

Top

Returns or sets the distance between the internal top edge of the control and the top edge of its container.

Value

Returns the underlying data value of the contents of the textbox portion of the control. It is not updated until the AfterUpdate event of the control has fired. The Value property behaves differently depending on the DataType properties. The following table describes this behaviour

DataType	Behaviour of the Value property
0-String 2-Long Integer 3-Integer 4-Byte	Same as the Text property. However, if the Text property returns an Empty String, the Value property returns Null.
1-Boolean	The Boolean constant True or False, or Null.
5-Decimal	The underlying number as a decimal data type, without any commas, percent signs, dollar signs, or other non-numeric characters. If DisplayFormat is Percent, Value returns the number as a decimal (i.e. if Text = 15.0%, Value returns 0.15).
6-Date	The underlying value as a Date data type, or Null.

The behaviour of the Value property when DataType is Boolean, Decimal, or Date is one of the most useful features of the Unicus Text Box. It allows you to display the data to the user in a way that is meaningful to them, while relying on the Value property to tell you exactly what the value truly is.

The Value property returns Null if the control is empty, whereas the Text property returns an Empty String. The Value property is also the Default property of the control, so when you refer to it in code, you don't have to explicitly use the syntax varValue = udsTextBox1.Value. You can simply use varValue = udsTextBox1.

Width

Returns or sets the overall width of the control.

WordWrap

Returns or sets a value that determines whether the label portion of the control expands vertically to fit the Caption text.

Custom Events

AfterUpdate

The AfterUpdate event fires after the text in the control has been validated to be appropriate for the specified DataType. Until the AfterUpdate event fires, the Value property is not updated, but the Text property is updated on the Change event of the control. If the BeforeUpdate event is cancelled, the AfterUpdate event will not fire, and the Value property will not be updated.

The control is updated, and the AfterUpdate event fires when the control loses focus, and when the user presses the Enter key (only if Multiline = False). It does not fire if you set the Text or Value properties in code.

BeforeUpdate (NewValue As Variant, Cancel As Boolean, RestoreOldValue As Boolean)

The BeforeUpdate fires when the user attempts to update the control by pressing Enter (when Multiline = False), or by attempting to leave the control. The data in the control is validated based on the DataType property. If the data is appropriate for the specified DataType, the Value property is updated and the AfterUpdate event is fired.

During the BeforeUpdate event, the Value property returns the old value of the control before the user made any changes. The NewValue parameter of the BeforeUpdate event returns the underlying value of the text that the user has typed. You can use this to perform your own validation based on your application's business rules. If NewValue fails your validation, you can set Cancel to True. This will prevent the control from being updated, the Value property will remain at its old value, and the control will not lose focus. However, the Text property will still be what the user entered, unless you set the RestoreOldValue parameter to True along with Cancel. This will cause the control to keep focus, and it will replace what the user typed with the value that the control held before the user started to change it. If Cancel is left False, then RestoreOldValue is ignored.

Change

The change event fires when the Text property of the textbox portion of the control changes. It has no parameters and cannot be cancelled. During this event, the Value property still returns the old value of the control.

Click(ClickLocationIsTextBox As Boolean)

The Click event fires when the user clicks either the caption or textbox portions of the control. The ClickLocationIsTextBox parameter tells you whether the caption or textbox portion of the control was clicked. The Click event does not fire if the user clicks in the dead area of the control. When the user clicks on the label portion of the control, the textbox automatically receives the focus. You do not have to do anything to make this happen. The Click event does not fire if the user clicks in the dead area of the control.

ControlError(ErrorNumber As Long, ErrorDescription As String, SupressMessage As Boolean)

The ControlError fires when certain errors occur in the control. ErrorNumber and ErrorDescription tell you what the error was. SuppressMessage allows you to prevent the default message from displaying so that you can either display your own message, or log the error, or do whatever you want with it. SuppressMessage defaults to False, so the default message will be displayed unless you explicitly set this parameter to True.

DblClick

The DblClick event fires when the user double-clicks in the textbox portion of the control. The DblClick event does not fire if the user clicks on the label portion or in the dead area of the control.

GotFocus

The GotFocus event fires when the textbox portion of the control receives the focus. Because the caption portion of the control is built from an intrinsic VB label control, it cannot receive the focus. However, if the user clicks on the caption portion of the control, the textbox portion of the control will automatically receive the focus.

Standard Events

Most other events that are standard to an intrinsic Visual Basic textbox are included with the Unicus Text Box. These include DragDrop, DragOver, KeyDown, KeyPress, KeyUp, LostFocus, MouseDown, MouseMove, MouseUp, Resize, and Validate.

Custom Methods

UpdateValue

You can call this method at any time in your code to force the control to update the Value property. This will cause the AfterUpdate event to fire but not the BeforeUpdate event. If you update the control using this method, it cannot be cancelled or reversed without writing your own code.