Introduction

Description

wesCommonLibrary is an Active X Automation server. It is intended to provide Visual Basic 4.0 developers commonly needed functionality beyond that provided by Visual Basic for Applications (VBA).

The library is provided in the form of a 32-bit dynamic link library (DLL). The server was developed in Visual Basic 4.0 and has been tested with Visual Basic 4.0 clients.

Classes

The library consists of two public classes:

Application
Library

Application objects are createable by the server's clients. Library objects are created through the Library method of the Application class.

Reference

Application Class

Properties
Methods

Application objects are createable by the server's clients. They are the only externally createable objects in the server and provide an entry point into the server.

Library objects are created through the Library method of the Application class. Once a client has created an Library object, it may release the Application object immediately if it will not need to create additional Library objects.

Application Class Properties

ClassName Property

Returns the class name of an object.

Syntax

object.ClassName

The ClassName property syntax has these parts:

Part	Description
object	Application object

Remarks

The ClassName property returns "Application" for an Application object and "Library" for an Library object.

Example

```
Debug.Print Library.ClassName

' displays: Library
```

LicenseNumber Property

See also:

UserName Property
Registration

Returns or sets the license number for a registered copy of the server.

Syntax

object.LicenseNumber [= number]

The License property syntax has these parts:

Part	Description
object	Application object
number	License number

Remarks

Registered users of server receive a license number and user name. Although unregistered copies of the server are fully functional, a modal message box is displayed each time a Library object is created. Setting the LicenseNumber and UserName properties with valid values prior to instantiating Library objects eliminates the modal message box.

The LicenseNumber property returns 0 until it has been set. Setting an invalid license number will not raise an error.

UserName Property

See also:

LicenseNumber Property
Registration

Returns or sets the user name for a registered copy of the server.

Syntax

object.UserName [= string]

The UserName property syntax has these parts:

Part	Description
object	Application object
string	User name

Remarks

Registered users of the server receive a license number and user name. Although unregistered copies of the server are fully functional, a modal message box is displayed each time an Error object is created. Setting the LicenseNumber and User Name properties with valid values prior to instantiating Error objects eliminates the modal message box.

The UserName property returns and empty string until it has been set. Setting an invalid user name will not raise an error.

Registration

LicenseNumber Property
UserName Property

wesCommonLibrary can be registered for US$29. Registration is available through several sources:

1. On CompuServe, use the Shareware Registration service (GO SWREG), Registration ID 14769. The registration fee is charged to your CompuServe account.

2. On the World Wide Web, set your browser to

    http://www.thecave.com/wesoft/

and follow the ordering directions. The registration fee can be charged to any major credit card via PsL, a credit card order service.

3. Complete the order form in Order.txt and fax it to PsL at 713-524-6398. The registration fee can be charged to any major credit card.

Important: These services are for REGISTRATION ONLY. Please see the Support section below for all other correspondence.

Registered users of wesCommonLibrary receive a license number and user name via electronic mail, or via postal mail if no electronic mail address is provided. Although unregistered copies of wesCommonLibrary are fully functional, a modal message box is displayed each time an Library object is instantiated. Setting the LicenseNumber and UserName properties of the Application object with valid values prior to instantiating Library objects eliminates the modal message box.

Version Property

Returns the version of the server.

Syntax

object.Version

The Version property syntax has these parts:

Part	Description
object	Application object

Remarks

The version is returned in the format:

Major.Minor.Revision

Application Class Methods

Library Method

Library Method

See also:

LicenseNumber Property
UserName Property
Registration

Returns a new instance of a Library object.

Syntax

object.Library

The Library method syntax has these parts:

Part	Description
object	Application object

Remarks

Registered users of thje server receive a license number and user name. Although unregistered copies of the server are fully functional, a modal message box is displayed each time a Library object is created. Setting the LicenseNumber and UserName properties with valid values prior to instantiating Library objects eliminates the modal message box.

Library Class

Properties
Methods

Overview

The methods of the Library class provide the functionality of the Visual Basic 4.0 Common Library.

Typically, Visual Basic applications would create a single instance of the Library class. The Library object would either be public or a property of a public "system" object.

Examples

Public Library Object

A well-architected Visual Basic application should start with Sub Main. A simple main code module could have the following structure:

```
Option Explicit

' public variables
Public Library as wesCommonLibrary.Library

Public Sub Main()

   ' the application starts here

   ' display the splash screen

   ' create the public library object
   Dim oApp as wesCommonLibrary.Application
   Set oApp = New wesCommonLibrary.Application
   oApp.UserName = "User Name"
   oApp.LicenseNumber = 1234567890
   Set Library = oApp.Library
   Set oApp = Nothing

   ' perform other startup activities here

   ' hide the splash screen and
   ' show the main form here

End Sub

Public Sub Shutdown()

   ' the application ends here when
   ' called by the Unload event
   ' of the main form

   ' destroy the public library object
   Set Library = Nothing

   ' do other shutdown activities here

End Sub
```

The library could then be called in all classes, forms, and code modules in the application with the syntax:

```
   Library.methodname
```

Library Object as Property of Public "System" Object

Using a public "Application" object, a simple main code module could have the following structure:

```
Option Explicit

' public variables
Public Application as Application

Public Sub Main()

   ' the application starts here

   ' display the splash screen

   ' create the public Application object
   Set Application = New Application

   ' perform other startup activities here

   ' hide the splash screen and
   ' show the main form here

End Sub

Public Sub Shutdown()

   ' the application ends here when
   ' called by the Unload event
   ' of the main form

   ' destroy the public Application object
   Set Application = Nothing

   ' do other shutdown activities here

End Sub
```

A simple class module for the Application class could have the following structure:

```
Option Explicit

' public properties
Public Library as wesCommonLibrary.Library

Private Sub Class_Initialize()

   ' create the public library object
   Dim oApp as wesCommonLibrary.Application
   Set oApp = New wesCommonLibrary.Application
   oApp.UserName = "User Name"
   oApp.LicenseNumber = 1234567890
   Set Library = oApp.Library
   Set oApp = Nothing

   ' perform other class initialization
   ' activities here

End Sub

Private Sub Class_Terminate()

   ' destroy the public library object
   Set Library = Nothing

   ' do other class destruction
   ' activities here

End Sub
```

The library could then be called in all classes, forms, and code modules in the application with the syntax:

```
   System.Library.methodname
```

Library Class Properties

ClassName Property

Library Class Methods

Assert
BooleanToYesNo
CenterWindow
ChangeDriveDir
ClearMaskedText
ClipCursor
ConvertSeconds
CreateDirectory
DelTree
DoubleQuoteString
DoubleSingleQuotes
GetToken
HiWord
IsDateEntry
IsLeapYear
IsNumericEntry
LengthLongestWord
ListComboIndexByItemData
ListIndexByText
LongestWord
LoWord
MakeDWord
Max
Min
MultiListAnySelected
MultiListRemoveSelected
MultiListSelectAll
MultiListUnselectAll
NullToNumeric
NullToString
NullTrim
OutlineItemByText
PadLeft
PadRight
ParseDate
ParsePath
QuoteString
ReplaceString
RInstr
Round
SelectText
SetRedraw
SetTopMost
SmartBeep
SplitName
Swap
UniqueFileName
UnloadAllForms

WindowsPath
WindowsSystemPath
YesNoToBoolean

Assert Method

Tests an assertion, displaying a message box if it fails.

Syntax

object.Assert Condition, Message

The Assert method has these parts:

Part	Description
object	Library object
Condition	Boolean expression representing condition
Message	String message

Remarks

Assertions are usually used in conditional code during debugging. The message can be either a statement of the condition or an exception statement.

Example

```
#If DebugMode Then
   Library.Assert nHeight > 0, "Height is positive."
#End If
```

BooleanToYesNo Method

See also:

YesNoToBoolean Method

Returns a "Yes" or "No" string equivalent to a Boolean value.

Syntax

object.BooleanToYesNo(Value)

The BooleanToYesNo method has these parts:

Part	Description
object	Library object
Value	Boolean expression

Remarks

BooleanToYesNo determines the string equivalent based on the following table:

Value	Returns
True	Yes
False	No

Example

```
Debug.Print _
   Library.BooleanToYesNo(True)

' displays: Yes
```

YesNoToBoolean Method

See also:

BooleanToYesNo Method

Returns the Boolean equivalent of a string expression.

Syntax

object.YesNoToBoolean(Value)

The YesNoToBoolean method has these parts:

Part	Description
object	Library object
Value	String expression

Remarks

YesNoToBoolean determines the boolean equivalent based on the following table:

Expression	Return Value
N	False
No	False
Y	True
Yes	True

False is returned for any string expression not found in the table. The expression check is not case sensitive.

Example

```
Debug.Print _
   Library.YesNoToBoolean("Yes")

' displays: True
```

CenterWindow Method

Centers a window within another window or within the screen.

Syntax

object.CenterWindow ChildWindow[, ParentWindow]

The CenterWindow method has these parts:

```
Part          Description

object         Library object
ChildWindow    Required. Object (window) to be centered
ParentWindow   Optional. Object (window) on which Child
centering will be based
```

Remarks

The child window will will be centered over the parent window if a parent window is specified. It will be centered on the scrren otherwise.

The left border of the child window will be placed on the left edge of the screen if centering it would cause the left border of the child to be off the screen. The same adjustment applies to the top border of the form and the top edge of the screen.

Both theChildWindow and ParentWindow objects must have Left, Top, Width, and Height properties. The ChildWindow object must have a Move method.

Example

```
Dim frmParent As Form1
Set frmParent = New Form1
frmParent.Show

Dim frmChild As Form2
Set frmChild = New Form2

Library.CenterWindow frmChild, _
   frmParent

frmChild.Show
```

ChangeDriveDir Method

Changes the default drive and directory (folder).

Syntax

object.ChangeDriveDir Path

The ChangeDriveDir method has these parts:

Part	Description
object	Library object
Path	String path

Remarks

The ChangeDriveDir method has the same result as using the ChDrive statement, followed by the ChDir statement.

Example

```
Library.ChangeDriveDir "c:\temp"
Debug.Print CurDir

' returns: C:\TEMP
```

ClearMaskedText Method

Clears the text from a Microsoft MaskedEditBox control.

Syntax

object.ClearMaskedText MaskedEditBox

The ClearMaskedText method has these parts:

Part	Description
object	Library object
MaskedEditBox	Microsoft MaskedEditBox control

Remarks

The ClearMaskedText method clears the text from the control, preserving the mask.

Example

```
Library.ClearMaskedText mskStartDate
```

ClipCursor Method

Restricts cursor to the confines of a form or control or releases previous restriction.

Syntax

object.ClipCursor [Window]

The ClipCursor method has these parts:

Part	Description
object	Library object
Window	Optional. Form or control

Remarks

Calling the ClipCursor method with the optional argument restricts the cursor to the confines of the specified form or control. Calling the method without an agrument releases the restriction.

The ClipCursor method does not work with "lightweight" VB controls such as Labels.

Example

```
Library.ClipCursor frmGotcha
```

ConvertSeconds Method

Converts a specified number of seconds into days, hours, minutes, and seconds.

Syntax

object.ConvertSeconds TotalSeconds, Seconds, Minutes[, Hours[, Days]]

The ConvertSeconds method has these parts:

Part	Description
object	Library object
TotalSeconds	Required. Long expression equalling number of seconds
Seconds	Required. Integer converted seconds
Minutes	Required. Integer converted minutes
Hours	Optional. Integer converted hours
Days	Optional. Integer converted days

Remarks

The maximum return value for the Seconds and Minutes arguments is 60.
The ConvertSeconds method will discard excess seconds if the Hours or Days arguments are not provided.

Example

```
Dim nSeconds As Integer
Dim nMinutes As Integer
Dim nHours As Integer
Dim nDays As Integer
Library.ConvertSeconds _
   12345, nSeconds, nMinutes, nHours, nDays
Debug.Print nSeconds, nMinutes, nHours, nDays

' displays: 45        25        3        0
```

CreateDirectory Method

Creates a directory (folder).

Syntax

object.CreateDirectory(Directory)

The CreateDirectory method has these parts:

Part	Description
object	Library object
Directory	String directory name

Remarks

CreateDirectory creates a directory only if it does not exist, avoiding the error from MkDir.

Example

```
Library.CreateDirectory "c:\test"
```

DelTree Method

Deletes a directory (folder) and all of its contents.

Syntax

object.Deltree Directory

The DelTree method has these parts:

Part	Description
object	Library object
Directory	String directory name

Remarks

The DelTree method works the same as the DOS DelTree command.

Example

```
Library.DelTree "C;\junk"
```

DoubleQuoteString Method

See also:

QuoteString Method

Returns a string, based on a source string surrounded with the double quote character.

Syntax

object.DoubleQuoteString(Source)

The DoubleQuoteString method has these parts:

Part	Description
object	Library object
Source	String expression to be enclosed in double quotes

Remarks

The source string is not changed.

Example

```
Dim sName As String
sName = "Fred"

Debug.Print _
   Library.DblQuoteString(sName)

' displays: "Fred"
```

QuoteString Method

See also:

DoubleQuoteString Method

Returns a string, based on a source string surrounded with the single quote character.

Syntax

object.QuoteString(Source)

The QuoteString method has these arguments:

Part	Description
object	Library object
Source	String expression

Remarks

The source string is not changed.

Example

```
Dim sName As String
sName = "Fred"

Debug.Print _
   Library.QuoteString(sName)

' displays: 'Fred'
```

DoubleSingleQuotes Method

Returns a string, based on a source string with single quotes replaced with two single quotes.

Syntax

object.DoubleSingleQuote(Source)

The DoubleSingleQuote method has these parts:

Part	Description
object	Library object
Source	String expression

Remarks

This method is useful for preparing strings for use in SQL statements. The source string is not changed.

Example

```
Debug.Print _
   Library.DoubleSingleQuote("He's back!")

' displays: He''s back!
```

GetToken Method

Returns the first token from the source string.

Syntax

object.GetToken(Source, Delimiter)

The GetToken method has these parts:

Part	Description
object	Library object
Source	Source string
Delimiter	String delimiter expression

Remarks

The token and the delimiter are stripped from the source in the process.

Example

```
Dim sSource As String
sSource = "Just do it"
Dim sToken As String
sToken = Library.GetToken(sSource, " ")
Debug.Print sSource, sToken

' displays: do it         Just
```

HiWord, LoWord Methods

See also:

MakeDWord Method

Returns the integer "high word" or "low word" portion of a long "double word."

Syntax

object.HiWord(DWord)

object.LoWord(DWord)

The HiWord and LoWord methods have these parts:

Part	Description
object	Library object
DWord	Long value

Remarks

The HiWord and LoWord methods correctly handles all Visual Basic integers (-32768 to 32767).

Example

```
Debug.Print Library.HiWord(1234567890),_
   Library.LoWord(1234567890)

' displays: 18838        722
```

MakeDWord Method

See also:

HiWord, LoWord Methods

Returns the long "double word" created from integer "high word" and "low word" values.

Syntax

object.MakeDWord(LoWord, HiWord)

The MakeDWord method has these parts:

Part	Description
object	Library object
LoWord	Integer value
HiWord	Integer value

Remarks

TheMakeDWord method correctly handles all Visual Basic integers (-32768 to 32767).

Example

```
Debug.Print Library.MakeDWord(722, 18838)

' displays: 1234567890
```

IsDateEntry Method

See also:

IsNumericEntry Method

Returns a Boolean value indicating if a key press is valid for a date entry.

Syntax

object.IsDateEntry(KeyAscii)

The IsDateEntry method has these parts:

Part	Description
object	Library object
KeyAscii	Integer key press value

Remarks

Valid key presses are digits, slash, and navigation keys. IsDateEntry does not check for valid date formats and does not validate the entry as a date ("99/99/99" will be accepted).

Example

```
' form module
Private Sub Text1_KeyPress(KeyAscii As Integer)
   If Not Library.IsDateEntry(KeyAscii) Then
      KeyAscii = 0
      Beep
   End If
End Sub
```

IsNumericEntry Method

See also:

IsDateEntry Method

Returns a Boolean value indicating if a key press is valid for a numeric entry.

Syntax

object.IsNumericEntry(KeyAscii)

The IsNumericEntry method has these parts:

Part	Description
object	Library object
KeyAscii	Integer key press value

Remarks

Valid key presses are digits, decimal, plus, minus, and navigation keys.

Example

```
' form module
Private Sub Text1_KeyPress(KeyAscii As Integer)
   If Not Library.IsNumericEntry(KeyAscii) Then
      KeyAscii = 0
      Beep
   End If
End Sub
```

IsLeapYear Method

Returns a Boolean value indicating whether a year is a leap year.

Syntax

object.IsLeapYear(Year)

The IsLeapYear method has these parts:

Part	Description
object	Library object
Year	Four-digit integer year

Remarks

The Year argument must be a four-digit year, such as 1996.

Example

```
Debug.Print Library.IsLeapYear(1996)

' displays: True
```

LengthLongestWord Method

See also:

LongestWord Method

Returns the integer length of the longest word in a string.

Syntax

object.LengthLongestWord(Source[, Delimiter])

The LengthLongestWord method has these parts:

Part	Description
object	Library object
Source	Required. String expression
Delimiter	Optional. Single character used to delimit words

Remarks

The LengthLongestWord method returns 0 for empty strings. The delimiter defaults to a space if it is not specified.

Example

```
Debug.Print _
   Library.LengthLongestWord("This is a test.")

' displays: 4
```

LongestWord Method

See also:

LengthLongestWord Method

Returns a string representing the longest word in a source string.

Syntax

object.LongestWord(Source[, Delimiter])

The LongestWord method has these parts:

Part	Description
object	Library object
Source	Required. String expression
Delimiter	Optional. Single character used to delimit words

Remarks

The LongestWord method returns an empty string for empty source strings. If there is a tie for the longest word, the word closest to the beginning of the source string is returned. The delimiter defaults to a space if it is not specified.

Example

```
Debug.Print _
   Library.LongestWord("This is a test.")

' displays: This
```

ListComboIndexByItemData Method

See also:

ListIndexByText Method
OutlineItemByText Method

Returns the ListIndex value for the first item in a ListBox or ComboBox control with its ItemData property equal to the specified value.

Syntax

object.ListComboIndexByItemData(ListCombo, Value)

The ListComboIndexByItemData method has these parts:

Part	Description
object	Library object
ListCombo	Object (ListBox or ComboBox control)
Value	Long value

Remarks

The ListCombo object must have an ItemData collection and a ListCount property. The ListComboIndexByItemData will return the index in the ItemData collection containing the value to find. If no match is found, -1 is returned.

Example

```
Form1.List1.AddItem "Fred"
Form1.List1.ItemData(Form1.List1.NewIndex) = 1234
Form1.List1.AddItem "Barney"
Form1.List1.ItemData(Form1.List1.NewIndex) = 2345
Form1.List1.AddItem "Wilma"
Form1.List1.ItemData(Form1.List1.NewIndex) = 3456
Form1.List1.AddItem "Betty"
Form1.List1.ItemData(Form1.List1.NewIndex) = 4567

Debug.Print _          Library.ListComboIndexByItemData(Form1.List1, 1234)

' displays: 2
nPatientIndex = _
   Library.ListComboIndexByItemData(lstPatients, 123456)
lstPatients.ListIndex = nPatientIndex
```

ListIndexByText Method

See also:

ListComboIndexByItemData Method
OutlineItemByText Method

Returns the ListIndex value for the first item in ListBox control with its Text property equal to the specified value.

Syntax

object.ListIndexByText(ListBox, Text[, Start])

The ListIndexByText method has these parts:

Part	Description
object	Library object
ListBox	Required. Object (ListBox control)
Text	Required. String expression to find
Start	Optional. Integer value specifying the ListIndex property of the starting item

Remarks

The ListIndexByText method returns -1 if the Text value is not found. The search starts at the first item is the Start argument is missing. The search stops when the first instance of Text is found. The search is case sensitive and considers trailing spaces.

Example

```
Dim nPatientIndex as Integer
nPatientIndex = _
   Library.ListIndexByText(lstPatients, "Fred")
lstPatients.ListIndex = nPatientIndex
```

OutlineItemByText Method

See also:

ListIndexByText Method
ListComboIndexByItemData Method

Returns the ListIndex value for the first item in an Outline control with its Text property equal to the specified value.

Syntax

object.ListIndexByText(Outline, Text[, Start])

The OutlineItemByText method has these parts:

Part	Description
object	Library object
Outline	Required. Object (Outline control)
Text	Required. String expression to find
Start	Optional. Integer value specifying the ListIndex property of the starting item

Remarks

The Outline object must have a List collection and a ListCount property.

The OutlineItemByText method returns -1 if the Text value is not found. The search starts at the first item is the Start argument is missing. The search stops when the first instance of Text is found. The search is case sensitive and considers trailing spaces.

Example

```
Form1.Outline1.AddItem "Flintstone"
Form1.Outline1.AddItem "Fred"
Form1.Outline1.Indent(1) = 2
Form1.Outline1.AddItem "Wilma"
Form1.Outline1.Indent(2) = 2
Form1.Outline1.AddItem "Rubble"
Form1.Outline1.AddItem "Barney"
Form1.Outline1.Indent(4) = 2
Form1.Outline1.AddItem "Betty", 5
Form1.Outline1.Indent(5) = 2

Debug.Print _
   Library.FindOutlineItemByText(Form1.Outline1, _
   "Wilma")

' displays: 2
```

Max, Min Methods

Returns a variant equal to the maximum or minimum of two values.

Syntax

object.Min(FirstValue, SecondValue)
object.Max(FirstValue, SecondValue)

The Max and Min methods have these parts:

Part	Description
object	Library object
FirstValue	Variant expression compared to SecondValue
SecondValue	Variant expression compared to FirstValue

Example

```
Debug.Print Library.Max(1, 2), _
   Library.Min(1, 2)

' displays: 2       1
```

MultiListAnySelected Method

See also:

MultiListRemoveSelected Method
MultiListSelectAll Method
MultiListUnselectAll Method

Returns a Boolean value indicating whether any items are selected in a multi-select list control.

Syntax

object.MultiListAnySelected(MultiSelectControl)

The MultiListAnySelected method has these parts:

Part	Description
object	Library object
MultiSelectControl	Object (multi-select list control)

Remarks

MultiListAnySelect will return True if any list items are selected. Otherwise, False is returned.

The MultiSelectControl object must have a Selected collection and a ListCount property.

Example

```
' form module
Option Explicit

Private Sub cmdRemove_Click()
   cmdRemove.Enabled = False
   cmdUnselectAll.Enabled = False
   Library.MultiListRemoveSelected lstPeople
   If lstPeople.ListCount = 0 Then
      cmdSelectAll.Enabled = False
   End If
End Sub

Private Sub cmdSelectAll_Click()
   Library.MultiListSelectAll lstPeople
End Sub

Private Sub cmdUnselectAll_Click()
   Library.MultiListUnselectAll lstPeople
End Sub

Private Sub Form_Load()
   lstPeople.AddItem "Fred"
   lstPeople.AddItem "Barney"
   lstPeople.AddItem "Wilma"
   lstPeople.AddItem "Betty"
   cmdRemove.Enabled = False
   cmdUnselectAll.Enabled = False
End Sub

Private Sub lstPeople_Click()
   If Library.MultiListAnySelected(lstPeople) Then
      cmdRemove.Enabled = True
      cmdUnselectAll.Enabled = True
   Else
      cmdRemove.Enabled = False
      cmdUnselectAll.Enabled = False
   End If
End Sub
```

MultiListRemoveSelected Method

See also:

MultiListAnySelected Method
MultiListSelectAll Method
MultiListUnselectAll Method

Removes any selected list items from a multi-select list control.

Syntax

object.MultiListRemoveSelected(MultiSelectControl)

The MultiListRemoveSelected method has these parts:

Part	Description
object	Library object
MultiSelectControl	Object (multi-select ListBox control)

Remarks

The MultiSelectControl object must have a Selected collection, a ListCount property, and a RemoveItem method.

Example

```
' form module
Option Explicit

Private Sub cmdRemove_Click()
   cmdRemove.Enabled = False
   cmdUnselectAll.Enabled = False
   Library.MultiListRemoveSelected lstPeople
   If lstPeople.ListCount = 0 Then
      cmdSelectAll.Enabled = False
   End If
End Sub

Private Sub cmdSelectAll_Click()
   Library.MultiListSelectAll lstPeople
End Sub

Private Sub cmdUnselectAll_Click()
   Library.MultiListUnselectAll lstPeople
End Sub

Private Sub Form_Load()
   lstPeople.AddItem "Fred"
   lstPeople.AddItem "Barney"
   lstPeople.AddItem "Wilma"
   lstPeople.AddItem "Betty"
   cmdRemove.Enabled = False
   cmdUnselectAll.Enabled = False
End Sub

Private Sub lstPeople_Click()
   If Library.MultiListAnySelected(lstPeople) Then
      cmdRemove.Enabled = True
      cmdUnselectAll.Enabled = True
   Else
      cmdRemove.Enabled = False
      cmdUnselectAll.Enabled = False
   End If
End Sub
```

MultiListSelectAll, MultiListUnselectAll Methods

See also:

MultiListAnySelected Method
MultiListRemoveSelected Method

Select or unselects all items in a multi-select list control.

Syntax

object.MultiListSelectAll(MultiSelectControl)

object.MultiListUnselectAll(MultiSelectControl)

The MultiListSelectAll and MultiListUnselectAll methods have these parts:

Part	Description
object	Library object
MultiSelectControl	Object (multi-select list control)

Remarks

The MultiSelectControl object must have a Selected collection and hWnd, ListCount, and TopIndex properties.

Example

```
' form module
Option Explicit

Private Sub cmdRemove_Click()
   cmdRemove.Enabled = False
   cmdUnselectAll.Enabled = False
   Library.MultiListRemoveSelected lstPeople
   If lstPeople.ListCount = 0 Then
      cmdSelectAll.Enabled = False
   End If
End Sub

Private Sub cmdSelectAll_Click()
   Library.MultiListSelectAll lstPeople
End Sub

Private Sub cmdUnselectAll_Click()
   Library.MultiListUnselectAll lstPeople
End Sub

Private Sub Form_Load()
   lstPeople.AddItem "Fred"
   lstPeople.AddItem "Barney"
   lstPeople.AddItem "Wilma"
   lstPeople.AddItem "Betty"
   cmdRemove.Enabled = False
   cmdUnselectAll.Enabled = False
End Sub

Private Sub lstPeople_Click()
   If Library.MultiListAnySelected(lstPeople) Then
      cmdRemove.Enabled = True
      cmdUnselectAll.Enabled = True
   Else
      cmdRemove.Enabled = False
      cmdUnselectAll.Enabled = False
   End If
End Sub
```

NullToNumeric Method

See also:

NullToString Method

Returns a numeric variant equal to the source value, if numeric, or 0 otherwise.

Syntax

object.NullToNumeric(Source)

The NullToNumeric method has these parts:

Part	Description
object	Library object
Source	Variant value

Remarks

NullToNumeric returns a numeric variant 0 if the sourve value is Null or any other non-numeric value. Otherwise, it returns the original value as a numeric variant.

Example

```
Dim vNull As Variant
vNull = Null

Debug.Print _
   Library.NullToNumeric(vNull)

' displays: 0
```

NullToString Method

See also:

NullToNumeric Method

Returns a string equal to the source value, if not Null, or an empty string otherwise.

Syntax

object.NullToString(Source)

The NullToString method has these parts:

Part	Description
object	Library object
Source	Variant value

Remarks

NullToString returns a empty string if the sourve value is Null. Otherwise, it returns the original value as a string.

Example

```
Dim vNull As Variant
vNull = Null

Debug.Print _
   "(" & Library.NullToString(vNull) & ")"

' displays: ()
```

NullTrim Method

Returns a string, based on a source string with all trailing Null characters (Chr$(0)) removed.

Syntax

object.NullTrim(Source)

The NullTrim method has these parts:

Part	Description
object	Library object
Source	String expression

Remarks

Embedded Null characters are not changed or removed. The source string is not changed.

Example

```
Dim sTest As String
sTest = "Fred" & Chr$(0)

Debug.Print _
   Len(Library.NullTrim(sTest))

' displays: 4
```

PadLeft, PadRight Methods

Return strings, based on a source string padded with a specified character to a specified length.

Syntax

object.PadLeft(Source, Length, [PadChar])
object.PadRight(Source, Length, [PadChar])

The PadLeft and PadRight methods have these parts:

Part	Description
object	Library object
Source	Required. String expression to be padded
Length	Required. Integer length of the padded string expression
PadChar	Optional. String expression whose first character is used to pad the return string

Remarks

PadLeft will pad the character(s) to the left, or beginning, of the source string. PadRight will pad the character(s) to the right, or end, of the source string. If the source string is longer than the specified length, the returned string is truncated on the appropriate side.

If PadChar is missing, the string will be padded with spaces (Chr$(32)).

The source string is not changed.

Example

```
Dim sTest As String
sTest = "Fred"

Debug.Print _
   Library.PadLeft(sTest, 10, "*")

' displays: ******Fred

Debug.Print _
   Library.PadRight(sTest, 10, "*")

' displays: Fred******
```

ParseDate Method

Parses a date expression into month, day, and year components.

Syntax

object.ParseDate Source, Month, Day, Year

The DateParse method has these parts:

Part	Description
object	Library object
Source	Variant expression representing a date
Month	Two-digit string expression
Day	Two-digit string expression
Year	Two-digit string expression

Example

```
Dim sMonth As String
Dim sDay As String
Dim sYear As String

Library.ParseDate "12/25/96", sMonth, _
   sDay, sYear
Debug.Print sMonth, sDay, sYear

' displays: 12           25           96
```

ParsePath Method

Parses a path into drive, folder (directory), and document (file) components.

Syntax

object.ParsePath Path, Drive, Folder, Document

The ParsePath method has these parts:

Part	Description
object	Library object
Path	String path
Drive	String drive
Folder	String folder
Document	String document

Example

```
Const TEST_PATH = "c:\windows\win.ini"

Dim sDrive as String
Dim sFolder as String
Dim sDocument as String

Library.ParsePath TEST_PATH, _
   sDrive, sFolder, sDocument
Debug.Print sDrive, sFolder, _
   sDocument

' displays: c:            \windows      win.ini
```

ReplaceString Method

Returns a string, based on a source string with all instances of one substring replaced with a second substring.

Syntax

object.ReplaceString(Source, CurrentString, NewString)

The ReplaceString method has these parts:

Part	Description
object	Library object
Source	String expression
CurrentString	String expression to be replaced
NewString	String expression that will replace CurrentString

Remarks

The source string is not changed.

Example

```
Dim sTest As String
sTest = "Fred, Fred, and more Fred"

Debug.Print _
   Library.ReplaceString(sTest, "Fred", _
   "Barney")

' displays: Barney, Barney, and more Barney
```

RInstr Method

Returns the position of the last occurrence of one string within another.

Syntax

object.RInstr(String1, String2 [, Start] [, Compare])

The RInstr method has these parts:

Part	Description
object	Library object
String1	Required. String expression being searched
String2	Required. String expression sought
Start	Optional. Numeric expression that sets the starting position for each search
Compare	Specifies the type of string comparison

Remarks

The search begins at the last character position if the Start argument is omitted.

The compare argument can be 0 or 1. Specify 0 to perform a binary comparison. Specify 1 to perform a textual, case-insensitive comparison. The default value is 0.

Example

```
Const TEST_SEARCH = "Mississippi"
Const TEST_SOUGHT = "s"

Debug.Print _
   Library.RInstr(TEST_SEARCH, _
   TEST_SOUGHT)

' displays: 7
```

Round Method

Returns a numeric variant, based on a source value rounded to a value with the specified number of decimal places.

Syntax

object.Round(Value, DecimalPlaces)

The Round method has these parts:

Part	Description
object	Library object
Value	Variant numeric value
DecimalPlaces	Integer number of decimal places

Remarks

Round returns an empty variant for non-numeric values.

Example

```
Dim fRoundDown As Double
fRoundDown = 0.134
Dim fRoundUp As Double
fRoundUp = 0.136

Debug.Print _
   Library.Round(fRoundDown, 2), _
   Library.Round(fRoundUp, 2)

' displays:  0.13          0.14
```

SelectText Method

Selects the text in a control.

Syntax

object.SelectText Textbox

The SelectText method has these parts:

Part	Description
object	Library object
Textbox	Object (Textbox control)

Remarks

The control must support the Text, SelStart, and SelLength properties.

Example

```
' form module
Option Explicit

Private Sub Text1_GotFocus()
   Library.SelectText Text1
End Sub
```

SetRedraw Method

Enables or disables the drawing of a form or control.

Syntax

object.SetRedraw Window, Redraw

The SetRedraw method has these parts:

Part	Description
object	Library object
Window	Object (form or control)
Redraw	Boolean value

Remarks

The object must have an hWnd property. Drawing is disabled if Redraw is False, and is enabled if Redraw is True.

Disabling drawing while a control such a list box or outline is being loaded will improve performance and eliminate flickering.

Example

```
' form module
Option Explicit

Private Sub cmdLoad_Click()
   Library.SetRedraw lstPeople, False
   Dim iPerson As Integer
   For iPerson = 1 To 1000
      lstPeople.AddItem "Person " & _
         CInt(iPerson)
   Next iPerson
   Library.SetRedraw lstPeople, True
End Sub
```

SetTopMost Method

Assigns or removes topmost status for a window.

Syntax

object.SetTopMost(Window, TopMost)

The SetTopMost method has these parts:

Part	Description
object	Library object
Window	Object (window) to make topmost
TopMost	Boolean value

Remarks

The window's topmost status will be set based on the value of TopMost. A window with topmost status will appear on top of all other windows in the same application, even if another window has focus.

The window object must have a hWnd property.

Example

```
' form module
Option Explicit

Private Sub cmdNoTopMost_Click()
   Library.SetTopMost Me, False
End Sub

Private Sub cmdTopMost_Click()
   Library.SetTopMost Me, True
End Sub
```

SmartBeep Method

Generates the specified sound.

Syntax

object.SmartBeep(SoundType)

The SmartBeep method has these parts:

Part	Description
object	Library object
SoundType	Integer expression

Remarks

The SmartBeep method handles the following sound types:

Constant	Value	Description
vbCritical	16	Critical Message icon.
vbQuestion	32	Warning Query icon.
vbExclamation	48	Warning Message icon.
vbInformation	64	Information Message icon.

The default system sound will be generated for any other value. Sounds are configured in the Sounds Control Panel applet.

Example

```
Library.SmartBeep vbExclamation
```

SplitName Method

Splits a single-word, aggregated name into components.

Syntax

object.SplitName(Source)

The SplitName method has these parts:

Part	Description
object	Library object
Source	String expression

Remarks

The SplitName method splits an aggregated name, such as variable names and database table and column names, into its component parts. Uppercase letters are used to identify the components. The components are return in a string separated by spaces.

Example

```
Debug.Print Library.SplitName("YabbaDabbaDo")

' displays: Yabba Dabba Do
```

Swap Method

Trades two values with one another.

Syntax

object.Swap FirstValue, SecondValue

The Swap method has these parts:

Part	Description
object	Library object
FirstValue	Variant value to trade with SecondValue
SecondValue	Variant value to trade with FirstValue

Remarks

In addition to trading values between variables, Swap can be also be used to trade values between the default properties of controls.

Example

```
Dim vFirst As Variant
vFirst = "First Value"
Dim vSecond As Variant
vSecond = "Second Value"

Library.Swap vFirst, vSecond
Debug.Print vFirst, vSecond

' displays: Second Value  First Value
```

UniqueFileName Method

Returns a string representing a unique file name (document), given a path and an extension.

Syntax

object.UniqueFileName(Path, Extension)

The UniqueFileName method has these parts:

Part	Description
object	Library object
Path	String path for file
Extension	String file extension

Example

```
Const TMP_PATH = "c:\temp"
Const TMP_EXT = "tmp"

Debug.Print _
   Library.UniqueFileName(TMP_PATH, _
   TMP_EXT)

' displays: c:\temp\15646.tmp
```

UnloadAllForms Method

Unloads all forms.

Syntax

object.UnloadAllForms(FormsCollection)

The UnloadAllForms method has these parts:

Part	Description
object	Library object
FormsCollection	Object (Forms collection)

Remarks

The UnloadAllForms method returns True if all forms were unloaded. It returns False, and stops unloading any subsequent forms, if a form cancels the unload message.

The UnloadAllForms method should not be called from code inside a form. It is intended to be used in object methods and property procedures.

Example

```
Library.UnloadAllForms Forms
```

WindowsPath Method

See also:

WindowsSystemPath Method

Returns the Windows path as a string.

Syntax

object.WindowsPath

The WindowsPath method has these parts:

Part	Description
object	Library object

Remarks

The string returned will not have a trailing backslash.

Example

```
Debug.Print Library.WindowsPath

' displays: C:\WINDOWS
```

WindowsSystemPath Method

See also:

WindowsPath Method

Returns the Windows system path as a string.

Syntax

object.WindowsSystemPath

The WindowsSystemPath method has these parts:

Part	Description
object	Library object

Remarks

The string returned will not have a trailing backslash.

Example

```
Debug.Print Library.WindowsSystemPath

' displays: C:\WINDOWS\SYSTEM
```

About

wesCommonLibrary Version 1.0.0
Written for West End Software, Inc.

History

Version 1.0.0 (3/3/97)

Initial release

License

wesCommonLibrary License Agreement

WEST END SOFTWARE LICENSE AGREEMENT AND LIMITED WARRANTY
West End Software, Inc. ("West End") grants you the right to use this West End software product ("Software"), including any accompanying documentation, in the manner provided below. This Software is owned by West End or its suppliers and is protected by copyright law and international copyright treaty.

The Software is a "shareware program." A shareware program is a program provided at no charge to the you for evaluation. You may share it with others, but you must not give it away altered or as part of another system.

If you find the Software useful and find that you are using the Software and continue to use the Software after a reasonable trial period, you must make a registration payment of US$29 to West End. 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 the Software must register and pay for their copies of the Software within 30 days of first use or their license is withdrawn. Site-License arrangements may be made by contacting West End.

Anyone distributing the Software for any kind of remuneration must first contact West End for authorization. This authorization will be automatically granted to distributors recognized by the (ASP) as adhering to its guidelines for shareware distributors, and such distributors may begin offering the Software immediately. However, West End must still be advised so that the distributor can be kept up-to-date with the latest version of the Software.

If you have purchased a license for the Software, you may transfer the license on a permanent basis provided you retain no copies of the Software and the recipient agrees to the terms of this license agreement. Except as provided in this agreement, you may not transfer, rent, lease, lend, copy, modify, translate, sublicense, time-share or electronically transmit or receive the Software or documentation. You acknowledge that the Software in source code form remains a confidential trade secret of West End and/or its suppliers and therefore you agree not to modify the Software or attempt to decipher, decompile, disassemble or reverse engineer the Software, except to the extent applicable laws specifically prohibit such restriction.

LIMITED WARRANTY
West End warrants that the Software, as updated and when properly used, will perform substantially in accordance with the accompanying documentation for a period of thirty (30) days from the date of registration. Any implied warranties on the Software are limited to thirty (30) days. Some states/jurisdictions do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.

West End's and its suppliers' entire liability and your exclusive remedy shall be, at West End's option, either (a) return of the price paid, or (b) repair or replacement of the Software that does not meet West End's Limited Warranty. This Limited Warranty is void if failure of the Software has resulted from accident, abuse, or misapplication. Any replacement Software will be warranted for the remainder of the original warranty

period.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, WEST END AND ITS SUPPLIERS DISCLAIM ALL OTHER WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WITH REGARD TO THE SOFTWARE AND THE ACCOMPANYING DOCUMENTATION. THIS LIMITED WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE OTHERS, WHICH VARY FROM STATE/JURISDICTION TO STATE/JURISDICTION.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL WEST END OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THIS WEST END PRODUCT, EVEN IF WEST END HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME STATES/JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

HIGH RISK ACTIVITIES
The Software is not fault-tolerant and is not designed, manufactured or intended for use or resale as on-line control equipment in hazardous environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines, or weapons systems, in which the failure of the Software
could lead directly to death, personal injury, or severe physical or environmental damage ("High Risk Activities"). West End and its suppliers specifically disclaim any express or implied warranty of fitness for High Risk Activities.

U.S. GOVERNMENT RESTRICTED RIGHTS
The Software and documentation are provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraphs (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs (c)(1) and (2) of the Commercial Computer Software-Restricted Rights at 48 CFR 52.227-19, as applicable. Manufacturer is West End Software, Inc.

GENERAL PROVISIONS
This agreement may only be modified in writing signed by you and an authorized officer of West End. If any provision of this agreement is found void or unenforceable, the remainder will remain valid and enforceable according to its terms. If any remedy provided is determined to have failed for its essential purpose, all limitations of liability and exclusions of damages set forth in the Limited Warranty shall remain in effect.

This agreement shall be construed, interpreted and governed by the laws of the State of Missouri, U.S.A. This agreement gives you specific legal rights; you may have others which vary from state to state and from country to country. West End reserves all rights not specifically granted in this agreement.

West End Software, Inc.
wesoft@thecave.com
wesoft@compuserve.com

Support

Questions, comments, and suggestions regarding wesCommonLibrary should be directed to West End Software, Inc.

CompuServe: Send email to wesoft
Internet: Send email to wesoft@thecave.com

Trademarks

West End Software

West End Software, Inc. is committed to providing the computing community, both end users and developers, with quality products.

Other West End products include:

wesError

An Active X server providing application developers with a standard means of handling errors trapped in a procedure, either displaying the error or raising the error for trapping and handling in the calling procedure.

wesObjectCache

An Active X server providing application developers with a means of improving performance by caching Active X objects for reuse, particularly objects that must retrieve their properties from a database, file, remote data source such as a World Wide Web site, etc.

Desktop Maid

A desktop utility for Microsoft Windows 95 that resides the the Taskbar tray, allowing you to hide or show items on your desktop. Desktop Maid also provides easy access to items on your desktop through a menu, even when they are not visible.

For the latest list of West End products, please visit our web site at

http://www.thecave.com/wesoft/

Thank you for your support!