



## Paradox Engine Configuration Utility not included with Visual Basic (Readme)

In the Crystal Reports for Visual Basic Help file, the topic "Paradox Engine Configuration Utility" refers to a file named PXENGCFG.EXE, which is used to configure the Paradox engine for a network. This file is not included with Visual Basic because the Microsoft Jet database engine handles all connections to Paradox data.

**For more information** To use the Jet database engine for connecting to Paradox data, see "Working With Paradox Data," in *Guide to Data Access*.

## ActiveX Controls that Disappear on HTML Pages (Readme)

Using HTML tools to insert certain ActiveX controls may cause the control not to appear when the HTML page is viewed in a browser. Controls for which this problem occurs include the MSChart and ImageList controls. Tools that create this problem include ActiveX Control Pad.

To remedy this, check the VBScript generated by the tool, and delete the Data attribute from the Object tag.

### ► To insert an MSChart Control using ActiveX Control Pad

- 1 Start ActiveX Control Pad..
- 2 On the **Edit** menu, click **Insert ActiveX Control**.
- 3 In the **Insert ActiveX Control** dialog box, double-click **Microsoft Chart Control**.
- 4 Close the **ActiveX Control Editor**.

When you close the editor, you will notice a large amount of text has been added to the HTML. Scroll up through the text until you find the beginning of the Object tag, which will resemble the text below:

```
<OBJECT ID="MSChart1" WIDTH=100 HEIGHT=51
  CLASSID="CLSID:31291E80-728C-11CF-93D5-0020AF99504A"
  DATA="DATA:application/x-
oleobject;BASE64,gB4pMYxyzxGT1QAgr5lQSiFDNBIIAAAVgoAACsFAAC9ZAYFAQAAAAAA
//// ...
```

- 5 To make the MSChart control display properly, you must delete all of the text for the Data attribute. When you do this, the HTML will resemble the code below:

```
<OBJECT ID="MSChart1" WIDTH=100 HEIGHT=51
  CLASSID="CLSID:31291E80-728C-11CF-93D5-0020AF99504A"
</OBJECT>
```

### Design time properties must be set at run time

Unfortunately, deleting the Data attribute also means you cannot set any design-time properties for the control, because the design-time properties have also been deleted. This means you must set all properties at run time. For controls such as the ImageList control, this means you cannot load pictures into the control at design time. Instead, you must use VBScript to load the pictures. Moreover, the LoadPicture function (needed to load graphics into the ImageList control) is not available in VBScript. To load pictures into the ImageList control, you must instead create your own control that exposes a public method. That method should then invoke the LoadPicture function.

## GetSelectedText Method Replaced by GetSelection Method (Readme)

The **GetSelectedText** method mentioned in the topic "Manipulating the IDE with Add-ins," in the *Component Tools Guide*, has been replaced by the **GetSelection** Method.

In the following sentence, found under the "Code Panel" heading, substitute "GetSelectedText" with "GetSelection":

"You can use the GetSelectedText method to copy selected code into the Windows clipboard."

## Upgrade ActiveX Controls Automatically from Project Properties Dialog (Readme)

The **Upgrade ActiveX Controls** option on the General Tab of the Project Properties dialog box is not adequately documented. In brief, the option allows you to specify if the project will upgrade controls automatically if new versions are available. If the option is checked (the default), ActiveX controls will be automatically updated if newer versions are present. If the option is unchecked, whenever the project is loaded, a dialog box will prompt you to upgrade the controls.

## Connect Event (Winsock Control)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmevtConnectEventWinsockC"}  
HLP95EN.DLL,DYNALINK,"Example":"rmevtConnectEventWinsockX":1}  
To":"rmevtConnectEventWinsockA;vbobjWinsockControl"} {ewc  
HLP95EN.DLL,DYNALINK,"Specifics":"rmevtConnectEventWinsockS"} {ewc  
{ewc HLP95EN.DLL,DYNALINK,"Applies
```

Occurs when a Connect operation is completed.

### Syntax

*object*.Connect()

The *object* placeholder represents an object expression that evaluates to a **Winsock** control.

### Remarks

Use the Connect event to confirm when a connection has been made successfully.

## Enterprise Sample Applications in Entpris, not Clisvr directory (Readme)

The Enterprise sample apps, shown in the list below, are found in the Entpris directory, instead of the clisvr directory, as stated in the sample.hlp.

### **Sample application name**

---

ape

callback

hello

interface

passthru

poolmgr

## Click Event (SSTab Control) Example has Syntax Error (Readme)

The example for the SSTab control's Click event has an error in the SaveSetting statement. The parentheses should be omitted. The correct syntax is shown below.

```
Private Sub sstbPrefs_Click(PreviousTab As Integer)
    Dim ThisSetting As String
    Select Case PreviousTab
        Case 0
            If optLoanLen(0) = True Then
                ThisSetting = "Months"
            Else
                ThisSetting = "Years"
            End If
            SaveSetting "LoanSheet", "LoanLength", _
                "Period", ThisSetting
        Case 1
            Dim X As Integer
            For X = 0 To 3
                If optPctsShown(X) = True Then
                    SaveSetting "LoanSheet", _
                        "InterestRate", _
                        "Precision", optPctsShown(X).Tag
                Exit For
            End If
        Next X
    End Select
End Sub
```



## MSComm Control Example Error (Readme)

The MSComm Control example has an error in the line demonstrating the Output and InBufferCount properties. The code in question is:

```
MSComm1.Output = "AT" + Chr$(13)
' Wait for data to come back to the serial port.

Do
    DoEvents
    Loop Until MSComm1.InBufferCount >= 2
    ' Read the "OK" response data in the serial port.
    Instring = MSComm1.Input
    ' Close the serial port.
    MSComm1.PortOpen = False
```

Although this can be replaced by the code below,

```
MSComm1.Output = "AT" + Chr$(13)
' Wait for data to come back to the serial port.

Do
    DoEvents
    Loop Until MSComm1.InBufferCount >= 7 ' changed
                                           ' value

    ' Read the "OK" response data in the serial port.
    Instring = MSComm1.Input
    ' Close the serial port.
    MSComm1.PortOpen = False
```

a better change would be as follows:

```
MSComm1.Output = "ATV1Q0" & Chr$(13) ' Ensure that
' the modem respond with "OK".
' Wait for data to come back to the serial port.
Do
    DoEvents
    Buffer$ = Buffer$ & MSComm1.Input
    Loop Until InStr(Buffer$, "OK" & vbCRLF)
    ' Read the "OK" response data in the serial port.
    ' Close the serial port.
    MSComm1.PortOpen = False
```

## BookSaleClient Sample: Sales Model button not depressed (Readme)

When the BookSaleClient sample application starts, none of the buttons in the Sales Model section is depressed. By default, however, the first Sales Model button is used when the Execute button is clicked. You can, of course, click any of the Sales Model buttons at any time to change the chart type.

## GetChunk Method, StateChanged Example Needs DoEvents (Readme)

The code example for the Internet Transfer Control's GetChunk method requires DoEvents statements inserted at the appropriate places. The corrected code is shown below:

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
    ' Retrieve server response using the GetChunk
    ' method when State = 12. This example assumes the
    ' data is text.

    Select Case State
        ' ... Other cases not shown.

    Case icResponseReceived ' 12
        Dim vtData As Variant ' Data variable.
        Dim strData As String: strData = ""
        Dim bDone As Boolean: bDone = False

        ' Get first chunk.
        vtData = Inet1.GetChunk(1024, icString)
        DoEvents

        Do While Not bDone

            strData = strData & vtData
            ' Get next chunk.
            vtData = Inet1.GetChunk(1024, icString)
            DoEvents

            If Len(vtData) = 0 Then
                bDone = True
            End If
        Loop

        txtData.Text = strData
    End Select

End Sub
```

## MSChart: Values for Updated Events (Readme)

The following events for the MSChart control have an *updateFlags* argument that is not entirely documented:

AxisLabelUpdated event

AxisTitleUpdate event

AxisUpdated event

ChartUpdated event

DataUpdated event

FootnoteUpdated event

LegendUpdated event

PlotUpdated event

PointLabelUpdated event

SeriesUpdated event

TitleUpdated event

Each of these events notifies the control when an aspect of the chart has been updated.

Consequently, the *updateFlags* argument specifies how much of the chart has been changed, from no change, to repositioning the major parts of the chart.

The constants for the *updateFlags* argument will not work. Instead, use the values listed in the table below.

Constant	Value
<b>VtChNoDisplay</b>	0
<b>VtChDisplayPlot</b>	1
<b>VtChLayoutPlot</b>	2
<b>VtChDisplayLegend</b>	4
<b>VtChLayoutLegend</b>	8
<b>VtChLayoutSeries</b>	16
<b>VtChPositionSection</b>	32

## Component Manager Not Included in VB5 (ReadMe)

The *Programmer's Guide* and the *Guide to Building Client/Server Applications with Visual Basic* both refer to the Component Manager. The Component Manager is not included in this version of Visual Basic.

## Chapter Title Changed in 'Guide to Data Access Objects' (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscWrongCrossRefInDach01csfC"}

The correct title of Chapter 8 in the *Guide to Data Access Objects* is "Creating Multiuser Applications". This chapter title is listed incorrectly in the first topic of Chapter 1, *Guide to Data Access Objects*.

## Feedback Web Page URL(ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscURLForVBFeedbackWebPageC"}

The correct location of the Visual Basic feedback web page is [www.microsoft.com/support/feedback/mswish.htm](http://www.microsoft.com/support/feedback/mswish.htm). The URL is incorrectly listed in the topic "Visual Basic Online Links" in Chapter 1 of the *Programmer's Guide*.

## SourceColumn Property is Read/Write at Run Time (Readme)

{ewc HLP95EN.DLL,DYNALINK,"See

Also":"rdproSourceColumnSourceTable;rmmscSourceColumnPropertyIsReadWriteAtRunTimeC"}

The SourceColumn Property of the rdoColumn Object is read/write at run time. The help topic "SourceColumn Property (Remote Data)" incorrectly says it is read-only at run time.



## Equi-Joins Example Code in 'Guide to Data Access Objects' (Readme)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscEquiJoinsExampleCodeInGuideToDataAccessObjectsC"}

Below is a corrected version of example code which appears in chapter 5, "Writing SQL Queries", under the heading "Inner Joins".

```
Dim MyQRY As QueryDef
MyQRY.SQL = "SELECT DISTINCTROW Sum([Unit Price] * " _
& "[Quantity]) AS [Sales], [First Name] & chr(32) & " _
& "[Last Name] AS Name FROM Employees " _
& "INNER JOIN (Orders INNER JOIN [Order Details] " _
& "ON Orders.[Order ID] = [Order Details].[Order ID]) " _
& "ON Employees.[Employee ID] = Orders.[Employee ID] " _
& "GROUP BY [First Name] & chr(32) & [Last Name];"
```

## Control Class (ReadMe)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmojbControlClassReadMeC"}           {ewc
HLP95EN.DLL,DYNALINK,"Example":"rmojbControlClassReadMeX":1}             {ewc
HLP95EN.DLL,DYNALINK,"Properties":"rmojbControlClassReadMeP"}           {ewc
HLP95EN.DLL,DYNALINK,"Methods":"rmojbControlClassReadMeM"}             {ewc
HLP95EN.DLL,DYNALINK,"Events":"rmojbControlClassReadMeE"}             {ewc
HLP95EN.DLL,DYNALINK,"Specifics":"rmojbControlClassReadMeS"}
```

Visual Basic provides the **Control** class as a generic variable type for controls. When you declare a variable **As Control**, you can assign it a reference to any control. You cannot create an instance of the **Control** class.

**Note** Accessing properties and methods of a control is faster if you use a variable declared with the same type as the control (for example, **As TreeView** or **As CommandButton**), because Visual Basic can use early binding. Visual Basic must use late binding to access properties and methods of a control assigned to a variable declared **As Control**.

## MaskColor Property (UserControl Object) (ReadMe)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmproNoHelpTopicForMaskColorPropertyOfUserControlC"} {ewc  
HLP95EN.DLL,DYNALINK,"Example":"rmproNoHelpTopicForMaskColorPropertyOfUserControlX":1} {ewc  
HLP95EN.DLL,DYNALINK,"Applies To":"rmproNoHelpTopicForMaskColorPropertyOfUserControlA"} {ewc  
HLP95EN.DLL,DYNALINK,"Specifics":"rmproNoHelpTopicForMaskColorPropertyOfUserControlS"}
```

Returns or sets the color that determines the transparent region of the bitmap assigned to the **MaskPicture** property of a **UserControl** object whose **BackStyle** property is set to 0 (Transparent).

### Syntax

*object*.**MaskColor** [= *color*]

The **MaskColor** property syntax has these parts:

Part	Description
<i>object</i>	An <u>object expression</u> that evaluates to an object in the Applies To list.
<i>color</i>	A value or <u>constant</u> that determines the color to be used as a mask, as described in Settings.

### Settings

Visual Basic uses the Microsoft Windows operating environment red-green-blue (RGB) color scheme. The settings for *color* are:

Setting	Description
RGB colors	Colors specified by using the Color palette or by using the <b>RGB</b> or <b>QBColor</b> functions in code.

### Remarks

When a bitmap is assigned to the **MaskPicture** property of a **UserControl** whose **BackStyle** property is set to 0 (Transparent), the control becomes transparent wherever it is covered by areas of the bitmap that match the **MaskColor** property.

Mouse events that occur over the transparent areas are received by the container or by controls that would otherwise be covered by the **UserControl**.

If there is no bitmap assigned to the **MaskPicture** property, or if the **BackStyle** property of the **UserControl** is not set to 0 (Transparent), then setting the **MaskColor** property has no effect.

For further details, see the **MaskPicture** property of the **UserControl** object.

**Note** Although **MaskColor** accepts the system color constants listed in the Visual Basic (VB) object library in the Object Browser, as described in Help for the **BackColor** and **ForeColor** properties, this is only useful if the **MaskPicture** bitmap contains a system color.

# MaskPicture Property (UserControl Object) (ReadMe)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmproNoHelpTopicForMaskPicturePropertyOfUserControlC"} {ewc  
HLP95EN.DLL,DYNALINK,"Example":"rmproNoHelpTopicForMaskPicturePropertyOfUserControlX":1} {ewc  
HLP95EN.DLL,DYNALINK,"Applies To":"rmproNoHelpTopicForMaskPicturePropertyOfUserControlA"} {ewc  
HLP95EN.DLL,DYNALINK,"Specifics":"rmproNoHelpTopicForMaskPicturePropertyOfUserControlS"}
```

Returns or sets the bitmap that, combined with the **MaskColor** property, determines the transparent and visible regions of a **UserControl** object whose **BackStyle** property is set to 0 (Transparent).

## Syntax

*object.MaskPicture* [= *picture*]

The **MaskPicture** property syntax has these parts:

Part	Description
<i>object</i>	An <u>object expression</u> that evaluates to an object in the Applies To list.
<i>picture</i>	A graphic image of one of the types described in Settings.

## Settings

The settings for *picture* are:

Setting	Description
Nothing	(Default) No picture. At design time, highlight the property value and press the Delete key; "(None)" will appear in the <u>Properties window</u> . At run time, assign the return value of the <b>LoadPicture</b> function with no filename, or use the <b>Set</b> statement to assign the value <b>Nothing</b> to the property.
(Bitmap, DIB, GIF, or JPEG)	Specifies an image-type bitmap. At design time, you can use the Properties window to enter a <u>string expression</u> specifying a file containing a graphic. At run time, you can set this property using a <b>Picture</b> object, using another object's <b>Picture</b> property, or using the <b>LoadPicture</b> function on an image-type <u>bitmap</u> file.

**Important** This feature is supported only for image-type bitmaps, such as GIF, JPEG, and DIB. It is not supported for Windows metafiles, icons, or cursors.

## Remarks

When a bitmap is assigned to the **MaskPicture** property of a **UserControl** whose **BackStyle** property is set to 0 (Transparent), the control becomes transparent wherever it is covered by areas of the bitmap that match the **MaskColor** property.

Mouse events that occur over the transparent areas are received by the container or by controls that would otherwise be covered by the **UserControl**.

The non-transparent parts of the **MaskPicture** bitmap are painted with the color specified by the **UserControl**'s **BackColor** property. These areas, which need not be contiguous, define the clipping region for drawing on the **UserControl**. That is, any drawing that is done on the surface of the **UserControl** is clipped to the non-transparent parts of the **MaskPicture** bitmap.

**Important** The bitmap assigned to the **MaskPicture** property is only used to define the transparent region and clipping region for the **UserControl**. The bitmap is never displayed. To display a bitmap on top of the clipping region defined by **MaskPicture** and **MaskColor**, you can assign the bitmap to the

**Picture** property of the **UserControl**, or draw it on the **UserControl** using **PaintPicture** in the **UserControl**'s Paint event.

You can create an animated clipping region for your control by drawing on a hidden **PictureBox**, and transferring the resulting image to the **MaskPicture** property of the **UserControl**.

**Note** When you set the **MaskPicture** property at design time, the graphic is saved and loaded with the .ctl and .ctx files that define the **UserControl**. If you make the project into a control component (.ocx file), the file contains the image. When you load a graphic at run time, by contrast, the graphic isn't saved with the component.

For more information, see "Giving Your Control a Transparent Background," in "Building ActiveX Controls" in *Creating ActiveX Components* in Books Online.

## ParentControlsType Property (ReadMe)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmproNoHelpTopicForParentControlsTypePropertyOfParentControlsC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"rmproNoHelpTopicForParentControlsTypePropertyOfParentControlsX":1}
{ewc HLP95EN.DLL,DYNALINK,"Applies To":"rmproNoHelpTopicForParentControlsTypePropertyOfParentControlsA"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"rmproNoHelpTopicForParentControlsTypePropertyOfParentControlsS"}
```

Returns or sets a value that determines whether the **ParentControls** collection contains references to controls incorporating the container's Extender object, or to controls without the Extender.

### Syntax

*object*.**ParentControlsType** [= *type*]

The **ParentControlsType** property syntax has these parts:

Part	Description
<i>object</i>	An <u>object expression</u> that evaluates to an object in the Applies To list.
<i>type</i>	An integer or named constant that indicates what the ParentControls collection will return.

### Settings

The settings for *type* are as follows:

Constant	Value	Description
<b>vbExtender</b>	0	(Default) The ParentControls collection will return the control and extender.
<b>vbNoExtender</b>	1	The ParentControls collection will return the control itself, without the extender.

### Remarks

The **ParentControls** collection allows you to access the other controls on a container where your control has been sited. The default is for the references to these controls to include the Extender object properties and methods provided by the container.

Some containers, such as Internet Explorer, provide an extender object that cannot be used by Visual Basic. In such a container, Visual Basic will raise an error when you attempt to access the objects in **ParentControls** using the default settings.

You can access the controls on an HTML page in Internet Explorer by setting **ParentControlsType** to **vbNoExtender**, so that **ParentControls** will contain references to the controls themselves, without the extender properties and methods.

Because the property can be set at run time, you can switch back and forth between **vbExtender** and **vbNoExtender** depending on what container your control is sited on. If necessary, you can alternate between the settings in containers that support both.

## Parent Property (UserControl Object) (ReadMe)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmproNoHelpTopicForUserControlParentC"} {ewc  
HLP95EN.DLL,DYNALINK,"Example":"rmproNoHelpTopicForUserControlParentX":1} {ewc  
HLP95EN.DLL,DYNALINK,"Applies To":"rmproNoHelpTopicForUserControlParentA"} {ewc  
HLP95EN.DLL,DYNALINK,"Specifics":"rmproNoHelpTopicForUserControlParentS"}
```

Returns a reference to the container object on which the control is sited. Not available at design time and read-only at run time.

### Syntax

*object*.**Parent**

The **Parent** property syntax has these parts:

Part	Description
<i>object</i>	An <u>object expression</u> that evaluates to an object in the Applies To list.

### Remarks

The **Parent** property returns a reference to the container object even when the **UserControl's AmbientProperties** object does not provide a **Parent** property. You can use the **Parent** property of the **UserControl** object to access the container's object model.

By testing TypeName(Parent), you can determine what container your control is sited on.

- **Excel** returns the workbook.
- **Word** returns the document.
- **Powerpoint** returns the presentation.
- **VB4/VB5** returns the form.
- **Internet Explorer** returns an object whose Script property returns the IOMWindow object.

For example, in Internet Explorer, the following code will change the background color of the HTML page on which your control is located:

```
Parent.Script.get_document.bgColor = "Blue"
```

More information on the Internet Explorer Scripting Object Model can be found on Microsoft's Web site.

**Important** Always use *late binding* for calls to the Internet Explorer Scripting Object Model. Using early binding will almost certainly cause compatibility problems in the future, while late binding will always work. In other containers, you can use early binding.

## Current Public/Instancing property values are not valid in this type of project (Readme)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmsgCurrentPublicinstancingPropertyValuesAreNotValidC"} {ewc  
HLP95EN.DLL,DYNALINK,"Specifics":"rmmsgCurrentPublicinstancingPropertyValuesAreNotValidS"}
```

This error occurs when you add a file to an existing project, and that project type doesn't support the setting of the **Instancing** or **Public** property of the module the file contains.

For example, you will get this error if you take a class module whose **Instancing** property is set to MultiUse from an ActiveX Exe project, and add it to a Standard Exe project. This is because Standard Exe projects only support the value Private for the **Instancing** property.

Similarly, this error will occur if you add a **UserControl** whose **Public** property is **True** to a Standard Exe project, because controls in Standard Exe projects can only be private.

You don't need to take any action as a result of this error, because Visual Basic automatically resets the **Instancing** or **Public** property to a value supported by the project type.



## Deleting a property with Control Interface Wizard doesn't update PropertyPage (Readme)

{ewc HLP95EN.DLL,DYNALINK,"See

Also": "rmmscDeletingPropertyWithControlInterfaceWizardDoesntUpdatePropertyPageC"}

The ActiveX Control Interface Wizard can be used to add and delete properties in existing ActiveX controls. If a control has a property page that displays a particular property, and you use the wizard to remove the property declaration from the control, the PropertyPage object will not be updated to reflect this.

As a result, the next time the property page is run, an error will occur. To avoid this error, you must manually remove all references to the deleted property from the PropertyPage object that used to display it.

## Description of interfaces that can be used with Implements (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscDescriptionOfInterfacesThatCanBeUsedWithImplementsC"}

When you create interfaces for use with the Implements statement, you can use Visual Basic or any tool that creates type libraries, such as the MkTypLib utility or the Microsoft Interface Definition Language (IDL) compiler.

Most classes created in Visual Basic define interfaces that work with Implements. When you create an interface by defining a class in Visual Basic, simply make sure that none of the properties or methods have underscores in their names.

Interfaces created with tools other than Visual Basic must follow certain restrictions in order to work with Implements. The following list includes most of these restrictions.

- Interface methods cannot have underscores in their names.
- Only [in] and [in,out] params are allowed — [out] only params are not allowed, and [lcid] parameters are not allowed.
- Method return types must be HRESULT, in order for errors to be propagated. You will not see the HRESULT in Visual Basic, as it is translated into an exception (raised error). To create a method that will have a return type when used in Visual Basic code, you must use [out, retval] on the final parameter.
- Only Automation data types may be used:

VB	MIDL
Integer	short
Long	long
Single	float
Double	double
Byte	unsigned char
Boolean	boolean or VARIANT_BOOL
String	BSTR
Variant	VARIANT
Date	DATE
Currency	CURRENCY or CY
Object	IDispatch *
IUnknown	IUnknown

- SAFEARRAY parameters containing any of the simple data types from the list above are allowed.
- Enum parameters are allowed.
- Dispinterface interface pointers are allowed as parameters.
- Dual interface pointers are allowed as parameters.
- CoClass parameters are allowed.
- If you're creating a type library in order to make a system interface usable with Implements, you **must not** use the [oleautomation] or [dual] attributes. Type libraries must be registered before you can add them to the Visual Basic References dialog box, and registering a type library with the [oleautomation] attribute will overwrite information required to remote the system interface. THIS WILL CAUSE OTHER APPLICATIONS ON THE SYSTEM TO FAIL. The [dual] attribute must not be used because it implies [oleautomation].

**Note** It may be useful to specify [oleautomation] while creating the typelib, in order to enforce correct types, but the type library must be built without the attribute **before** you reference it through the Visual Basic References dialog box.

- Unsigned long and unsigned short parameters are not included in the data type table, and are not allowed.
- User-defined data types (structures) are not allowed as parameters.
- Interfaces must be based on IUnknown or IDispatch. The full vtable (after IUnknown/IDispatch) must be described in a single interface.
- Restricted vtable entries are ignored and do not prevent the Implements statement from working.
- Most pointers cannot be passed as [in] parameters. (For example, as a remoting optimization, a C++ interface can declare a parameter as [in] VARIANT\* pVar. This will not work with Implements.) An [in] parameter can be a BSTR, a pointer to an interface (for example, IDispatch\*), or a SAFEARRAY pointer (SAFEARRAYs are always passed as pointers). An [in,out] parameter can be a pointer to an Automation type, or a double pointer to an interface (for example, IDispatch\*\*). (Note that 'ByVal As String' in Visual Basic maps to [in] BSTR. You cannot use [in] BSTR\* with Visual Basic.)
- Implements does not work with dispinterfaces.

For more information, see "Polymorphism," in "Programming with Objects" in the *Visual Basic Programmer's Guide*, and also "Providing Polymorphism by Implementing Interfaces," in "General Principles of Component Design" in *Creating ActiveX Components* in the *Component Tools Guide*. These topics can be found in Books Online.

## Errata: Conditional Compilation Code Example (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscErrataConditionalCompilationCodeExampleC"}

In "Using Conditional Compilation," in "More About Programming" in the *Visual Basic Programmer's Guide*, the example code is incorrect. The first **#Else** should be a **#ElseIf**:

```
#If conFrenchVersion Then
    ' <code specific to the French language version>.
#ElseIf conGermanVersion then
    ' <code specific to the German language version>.
#Else
    ' <code specific to other versions>.
#End If
```

## Error: 'Item' could not be loaded (ReadMe)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmsgErrorItemCouldNotBeLoadedC"} {ewc  
HLP95EN.DLL,DYNALINK,"Specifics":"rmmsgErrorItemCouldNotBeLoadedS"}
```

In addition to other causes, this error occurs when you attempt to add a file containing an existing Form, MDIForm, UserControl, PropertyPage, or UserDocument to a project in which you have selected the Unattended Execution option (accessed from the General tab of the Project Properties dialog box).

A project marked for unattended execution cannot support any form of user interaction, so the module types it can contain are severely restricted.

For more information, see "Scalability and Multithreading," in "Building Code Components" in *Creating ActiveX Components* in Books Online.

## Instancing property default varies depending on project type (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscInstancingPropertyDefaultIsMultiUseForActiveXExeDLLProjectsC"}

Online Help states that the default value of the **Instancing** property is Private. This is only true for class modules in Standard Exe projects.

When you insert a new class module into an ActiveX Exe project or an ActiveX DLL project, the default value of the **Instancing** property is MultiUse. When you insert a new class module into an ActiveX Control project, the default value of the **Instancing** property is PublicNotCreatable.

## Modal dialog or message box blocks events when debugging (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscModalDialogOrMessageBoxBlocksEventsWhenDebuggingC"}

The development environment cannot raise events while a modal form or message box is displayed, because of potential conflicts in the debugger. Therefore, events are suppressed until the modal form or message box is dismissed.

**Important** Suppression of events only happens in the development environment. Once a project is compiled, events will be raised even when a modal form or message box is displayed.

Example scenarios in which this can occur:

- A form with a **Timer** control on it is running in the development environment. Selecting Options from the Tools menu will open the Options dialog box, which is modal. Until the dialog is dismissed, the **Timer** control's Timer event will not be raised.
- An instance of a **UserControl** with a **Timer** control on it is placed on a form at design time. (The timer may be used to make the control appear animated; this effect can occur even in design mode, because controls can execute code at design time.) Selecting Add Class Module from the Project menu will open the Add Class Module dialog, which is modal. The **Timer** control's Timer event will be suppressed until the dialog is dismissed.
- A **UserDocument** contains a Timer control, and a command button that displays a message box. If the **UserDocument** is being debugged using Internet Explorer, pressing the button to display the message box will cause the **Timer** control's Timer event to be suppressed until the message box is dismissed.

## Projects compiled to Native Code still require MSVBVM50.DLL (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscProjectsCompiledToNativeCodeStillRequireMSVBVM50DLLC"}

All projects created with Visual Basic use the services of the run-time DLL (MSVBVM50.DLL). Among the services provided by this DLL are startup and shutdown code for your application, functionality for forms and intrinsic controls, and run-time functions like Format and CLng.

Compiling a project with the Native Code option means that the code you write will be fully compiled to the native instructions of the processor chip, instead of being compiled to p-code. This will greatly speed up loops and mathematical calculations, and may somewhat speed up calls to the services provided by MSVBVM50.DLL. However, it does not eliminate the need for the DLL.



## Require License Key only applies to ActiveX Control projects (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmdlgRequireLicenseKeyDisabledOnGeneralTabReadMeC"} {ewc  
HLP95EN.DLL,DYNALINK,"Specifics":"rmdlgRequireLicenseKeyDisabledOnGeneralTabReadMeS"}

Help for the General Tab on the Project Properties dialog box states that the Require License Key option "Enables licensing for a project that produces ActiveX Components (automation servers, user controls, and ActiveX controls."

This is incorrect. Licensing is only supported for ActiveX Control projects (that is, projects that compile to .ocx files). Therefore the option is disabled for all other project types.

## The VBIDE can't provide multiple instances of a SingleUse class (Readme)

```
{ewc HLP95EN.DLL,DYNALINK,"See  
Also":"rmmsgTheVisualBasicDevelopmentEnvironmentCantProvideMultipleInstancesC"} {ewc  
HLP95EN.DLL,DYNALINK,"Specifics":"rmmsgTheVisualBasicDevelopmentEnvironmentCantProvideMultipleInstancesS"}
```

The error occurs under the following conditions:

- A project that is running in the development environment contains a class whose Instancing property is set to SingleUse.
- A client project has previously created an instance of this SingleUse class.
- A client (it may be the same client, or another client) attempts to create a second instance of the SingleUse class.

The error occurs because a SingleUse object can only be created once per instance of the component that provides it. (That is, once an ActiveX Exe project is compiled, creating multiple instances of its SingleUse class will run multiple instances of the Exe component.) However, when the project is in the development environment, it is not possible to start a second instance of the project.

This is a restriction on debugging projects that provide SingleUse classes.

**Note** The error will occur even if the first client has released its instance of the SingleUse class. This is because a compiled Exe cannot provide a second instance of a SingleUse class, even after the first has been released.

For more information, see "Scalability Through Multiple Processes: SingleUse Objects," in "Building Code Components" in *Creating ActiveX Components* in the *Component Tools Guide* in Books Online.

## Version 'item3' of 'item1' is not registered. The control will be... (ReadMe)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmmsgVersion20OfitemIsNotRegisteredTheControlWillBeUpgradedC"}  
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"rmmmsgVersion20OfitemIsNotRegisteredTheControlWillBeUpgradedS"}
```

In addition to other causes, this message may occur from time to time when you are authoring ActiveX controls with Visual Basic. The message is harmless, requires no action on your part, and can be avoided as described below.

The message appears under the following circumstances:

- You compile your ActiveX control project to an .ocx file.
- You create a test project in another copy of Visual Basic and use the Controls tab of the Components dialog box (accessed from the Project menu) to select the .ocx file, which then appears in the Toolbar.
- You place an instance of a control on a form, and then save the test project.
- At some later date, you recompile the .ocx file.
- When you open the test project, the message will appear during project load.

The message occurs when using the default Version Compatibility setting (Project Compatibility, on the Components tab of the Project Properties dialog box). The Project Compatibility setting causes the type library version number of a control component (.ocx file) to be incremented each time you compile it. This is to allow easy upgrades of released controls.

Normally, when a new release of an .ocx file is installed on a developer's computer, it does not replace the old version (usually because the file names are different). When the developer opens an existing project that used the old version of the control, Visual Basic asks whether the control instances in the project should be upgraded to the new version.

In this case, there is no old version of the .ocx file, so Visual Basic automatically upgrades the controls, and provides a message informing you of this.

**Note** The type library version number is not in any way associated with the file version number you set using the Make tab of the Project Properties dialog box. The numbers displayed in this message are the decimal equivalents of the hexadecimal type library version numbers stored in the Windows Registry.

The message requires no action on your part in this scenario, because Visual Basic automatically incorporates the new type library version number into the test project.

You can avoid the message in several ways:

- Instead of compiling your .ocx file and testing it with a separate copy of Visual Basic, you can include both the ActiveX control project and the Standard Exe test project in a project group, and run them in the same copy of the development environment. This is described in "Creating an ActiveX Control," in *Creating ActiveX Components* in the *Component Tools Guide* in Books Online.
- Use the Component tab of the Project Properties dialog box to select Binary Compatibility with an earlier version of your .ocx file. The drawback to this is that if you later make a version incompatible change to your control (such as removing a property), you will get Version Compatibility warning messages when you compile.

**Note** If you're creating a new versions of an existing control, and you intend to make incompatible changes in the interface, this may not be your best option; Project Compatibility is better in that case, because it preserves the ability of developer's to upgrade the controls in their projects to your new version. You should change the file name, however, so that the new .ocx file doesn't overwrite the old.

For more information on type library versions and the Version Compatibility feature, see "Version Compatibility" in "Debugging, Testing, and Deploying Components" in *Creating ActiveX Components*

in the *Component Tools Guide* in Books Online.

## A property or method call included a reference to a private object. (Readme)

You included a reference to a private object as an argument or return value in a property or method call.

Private objects should never be passed outside a project. The following, all of which are prohibited, are possible causes for the error:

- A client invoked a property or method of an out-of-process component and attempted to pass a reference to a private object as one of the arguments. A client invoked a property or method of an out-of-process component and the component attempted to return a reference to a private object, or to assign such a reference to a **ByRef** argument.
- An out-of-process component has invoked a call-back method on its client and attempted to pass a reference to a private object
- An out-of-process component attempted to pass a reference to a private object as an argument of an event it was raising.
- A client attempted to assign a private object reference to a **ByRef** argument of an event it was handling.

**Notes** Although Visual Basic prevents you from passing references to private objects across processes, there are some cases in which Visual Basic can't detect this error and thus can't prevent it. Private objects are not designed to be used outside your project. If you pass them to a client, you may jeopardize program stability and cause incompatibility with future versions of Visual Basic. If you need to pass a private class of your own to a client, set the **Instancing** property to a value other than **Private**.

The error will always appear in the client, even if it is the fault of the server code.

For additional information, select the item in question and press F1.

Application not found. Looking for object with CLSID: {1}. (Readme)

Visual Basic could not find the application you are trying to use. This is a system error. If this problem persists, re-install Visual Basic or the affected application, and try again.

Application was launched but it didn't register a class factory.  
(Readme)

The application was opened but a class factory was not registered. This is a system error. If this problem persists, re-install Visual Basic or the affected application, and try again.

Class is not registered. Looking for object with CLSID: {1}. (Readme)

The class was not registered. This is a system error. If this problem persists, re-install Visual Basic and try again.



## Class not found. (Readme)

Visual Basic could not find the class that you specified. This is a system error. If this problem persists, re-install Visual Basic or the affected application, and try again.

Class object cannot be determined. Looking for object CLSID: {1.  
(Readme)

Visual Basic could not find the class object you indicated. This is a system error. If this problem persists, re-install Visual Basic or the affected application, and try again.

DLL for class not found. Looking for object with CLSID: {1.  
(Readme)

Visual Basic could not find the DLL for the class that you specified. This is a system error. If this problem persists, re-install Visual Basic or the affected application, and try again.

## Error in the DLL. Looking for object with CLSID: {1}. (Readme)

You have received a system error. If this problem persists, re-install Visual Basic or the affected application, and try again.

Interface not registered. Looking for object with CLSID: {1.  
(Readme)

The interface was not registered. This is a system error. If this problem persists, re-install Visual Basic or the affected application, and try again.

## Invalid class string. Looking for object with ProgID. (Readme)

The class string you are trying to use is invalid. This is a system error. If this problem persists, re-install Visual Basic or the affected application, and try again.

Object is not registered. Looking for object with CLSID: {1.  
(Readme)

The object you are trying to use is not registered. This is a system error. If this problem persists, re-install Visual Basic or the affected application, and try again.

## Unexpected error. (Readme)

You have received a system error. If this problem persists, re-install Visual Basic or the affected application, and try again.



## Using the Comment Block and Uncomment Block Commands

```
{ewc HLP95EN.DLL,DYNALINK,"See  
Also":"rmcmdUsingCommentUncommentCommandsC;vbcmdCommandBars;vbcmdCommentBlock;vbcmdEditCommandBar"}  
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"rmcmdUsingCommentUncommentCommandsS"}
```

The **Comment Block** and **Uncomment Block** commands can be found on the **Edit** toolbar. Use these commands to make a multiple lines of code into a comment block or to remove the comment characters from a block of code:

1. Add the **Edit** toolbar either by choosing the **Toolbars** command on the **Edit** menu and then choosing **Edit**; or by placing your cursor on the toolbar, clicking the right mouse button, and then choosing **Edit**.
1. Open a code module and highlight the code you want to use to create a comment block or the code from which you want to remove comment characters.
1. Click the **Comment Block** or **Uncomment Block** button on the **Edit** toolbar.

Either the code will be moved into a block and the comment characters added or it will be separated into individual pieces of code and the comment characters removed from it.

## Wrong OS or OS version for application. (Readme)

You are using an incompatible operating system or the wrong version of the operating system. Install a compatible operating system and try again.

## Controls Collection (Readme)

```
{ewc HLP95EN.DLL,DYNALINK,"See
Also":"vbcolControlsCollectionC;vbobjControlsC;vbproBooksOnlineJumpTopic;vbproControlsPropertyS"} {ewc
HLP95EN.DLL,DYNALINK,"Example":"vbcolControlsCollectionX":1} {ewc
HLP95EN.DLL,DYNALINK,"Properties":"vbcolControlsCollectionP"} {ewc
HLP95EN.DLL,DYNALINK,"Methods":"vbcolControlsCollectionM"} {ewc
HLP95EN.DLL,DYNALINK,"Events":"vbcolControlsCollectionE"} {ewc
HLP95EN.DLL,DYNALINK,"Specifics":"vbcolControlsCollectionS"}
```

A collection whose elements represent the controls on a container. The **Controls** collection has a **Count** property which specifies the number of controls in the collection, and an **Item** method which returns a member of the collection.

### Syntax

*object*.**Controls.Count**  
*object*.**Controls**(*index*)

The **Controls** collection syntax has these parts:

Part	Description
<i>object</i>	An <u>object expression</u> that evaluates to an object in the Applies To list.
<i>Index</i>	An integer with a range from 0 to <code>Controls.Count - 1</code> .

**Note** If the container is a Visual Basic module, such as a Form or UserControl, you don't have to supply the object expression when writing code within the module. If the container is a compiled ActiveX control, such as the ToolBar control, you must always supply the object expression.

### Remarks

The **Controls** collection enumerates loaded controls on a container. For example, you might use it to change the **BackColor** property of all the Label controls on a container.

You can use the **TypeOf** keyword with the **If** statement, or the **TypeName** function, to determine the type of a control in the **Controls** collection.

**Note** The **Controls** collection is not a member of the Visual Basic **Collection** class. It has a smaller set of properties and methods than a **Collection** object, and you cannot create instances of it.

## Error Constants (ComCtl32) (Readme)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstErrorConstantsComctl232ocxC;vbproBooksOnlineJumpTopic"}

Constant	Value	Description
cc2InvalidProcedureCall	5	Invalid procedure call
cc2BadFileNameOrNumber	&H34	Bad file name or number
cc2FileNotFound	&H35	File not found
cc2InvalidPropertyValue	&H17C	Invalid property value
cc2SetNotSupportedAtRuntime	&H17E	Property cannot be set at run time
cc2SetNotSupported	&H17F	Property is read-only
cc2InvalidObjectUse	&H1A9	Invalid object use
cc2WrongClipboardFormat	&H1CD	Specified format doesn't match format of data
cc2DataObjectLocked	&H2A0	DataObject formats list may not be cleared or expanded outside of the OLEStartDrag event
cc2ExpectedAnArgument	&H2A1	Expected at least one argument.
cc2RecursiveOleDrag	&H2A2	Illegal recursive invocation of OLE drag and drop
cc2FormatNotByteArray	&H2A3	Non-intrinsic OLE drag and drop formats used with SetData require Byte array data. GetData may return more bytes than were given to SetData.
cc2DataNotSetForFormat	&H2A4	Requested data was not supplied to the DataObject during the OLESetData event.
cc2InconsistentObject	&H8BA6	Internal state of the control has become corrupted
cc2ErrorDuringSet	&H8BA7	Unable to set property
cc2ErrorOpeningVideo	&H8BA8	Unable to open AVI file
cc2ErrorPlayingVideo	&H8BA9	Unable to play AVI file
cc2NoValidBuddyCtl	&H8BAA	BuddyControl property must be set first
cc2VideoNotOpen	&H8BAB	Must open AVI file first
cc2AutoBuddyNotSet	&H8BAC	AutoBuddy not set, no potential buddy controls found
cc2ErrorStoppingVideo	&H8BAD	Error trying to stop playing AVI file
cc2ErrorClosingVideo	&H8BAE	Error closing open AVI file
cc2CantStopAutoPlay	&H8BAF	Stop method does not effect AutoPlay property
cc2BuddyNotASibling	&H8BB0	BuddyControl must be a separate control within the same container
cc2NoUpDownAsBuddy	&H8BB1	An UpDown control cannot be buddied with another UpDown control

## LeftCol Property (Readme)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproLeftColPropertyActiveXControlsC"}

{ewc

```
HLP95EN.DLL,DYNALINK,"Example":"vbproLeftColPropertyActiveXControlsX":1} {ewc
HLP95EN.DLL,DYNALINK,"Applies To":"vbproLeftColPropertyActiveXControlsA"} {ewc
HLP95EN.DLL,DYNALINK,"Specifics":"vbproLeftColPropertyActiveXControlsS"}
```

Returns or sets an integer representing the leftmost visible column of a **DBGrid** control. this property is read-only at design time.

### Syntax

*object*.**LeftCol** [= *value*]

The **LeftCol** property syntax has these parts:

Part	Description
<i>object</i>	An <u>object expression</u> that evaluates to an object in the Applies To list.
<i>value</i>	A <u>numeric expression</u> indicating the leftmost visible column. The default value is 0.

## OLEDrop Constants (ComCtl32) (Readme)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstOLEDropConstantscomctl232ocxC;vbproBooksOnlineJumpTopic"}

Constant	Value	Description
cc2OLEDropEffectNone	0	No OLE drag/drop operation has taken place/would take place.
cc2OLEDropEffectCopy	1	A mask to indicate that a copy has taken place/would take place.
cc2OLEDropEffectMove	2	A mask to indicate that a move has take place/would take place.
cc2OLEDropEffectScroll	-2147483648 (&H80000000)	A mask to indicate that the drop target window has scrolled/would scroll.

## OLEDropEffect Constants (ComCtl32) (Readme)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcsOLEDropEffectConstantscomctl232ocxC;vbproBooksOnlineJumpTopic"}

Constant	Value	Description
cc2OLEDropManual	1	Accepts an OLE drag/drop under programmatic control only.
cc2OLEDropNone	0	Accepts no OLE drag/drop operations.

## VBComponents Property (Readme)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproVBComponentsPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproVBComponentsPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaobjVBProject;vbproVBComponentsPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproVBComponentsPropertyS"}

Returns a collection of the components contained in a project.

### Syntax

*object*.**VBComponents**

The **VBComponents** property syntax has these parts:

Part	Description
<i>object</i>	An <u>object expression</u> that evaluates to an object in the Applies To list.

### Remarks

Use the **VBComponents** collection to access, add, or remove components in a project. A component can be a form, module, or class. The **VBComponents** collection is a standard collection that can be used in a **For Each** block.

You can use the **Parent** property to return the project the **VBComponents** collection is in.

In Visual Basic for Applications, you can use **Import** method to add a component to a project from a file.

## VBNewProjects Collection (Readme)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcolVBNewProjectsCollectionC;vbproBooksOnlineJumpTopic"} {ewc
HLP95EN.DLL,DYNALINK,"Example":"vbcolVBNewProjectsCollectionX":1} {ewc
HLP95EN.DLL,DYNALINK,"Properties":"vbcolVBNewProjectsCollectionP"} {ewc
HLP95EN.DLL,DYNALINK,"Methods":"vbcolVBNewProjectsCollectionM"} {ewc
HLP95EN.DLL,DYNALINK,"Events":"vbcolVBNewProjectsCollectionE"} {ewc
HLP95EN.DLL,DYNALINK,"Specifics":"vbcolVBNewProjectsCollectionS"}
```

Represents all of the new projects in the development environment.

### Syntax

#### VBNewProjects

### Remarks

Use the **VBNewProjects** collection to access specific projects in an instance of the development environment. **VBNewProjects** is a standard collection that you can iterate through using a **For Each** block.



## Cannot delete multiple rows (DBGrid Control) (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjDBGrid"}

This error results from pressing the delete key when the SelBookmarks collection contains more than 1 bookmark.

## Caption text is too long (DBGrid Control) (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjDBGrid"}

This error occurs when the caption property or columns heading property is set with text over 255 characters in length.

## Column not found (DBGrid Control) (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjDBGrid"}

This error occurs when accessing the columns collection using an invalid column name.

## Control not properly initialized (DBGrid Control) (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjDBGrid"}

This error occurs during normal program execution if the grid's hwnd or ICursor has been destroyed.

## Data access error (DBGrid Control) (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjDBGrid"}

This is a general ICursor error. It occurs when accessing the data source if the data source has not provided additional information via the ISupportErrorInfo and IErrorInfo interfaces.

## Invalid bookmark (DBGrid Control) (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjDBGrid"}

This error results from setting a grid property or collection that requires a bookmark when the bookmark is invalid.

## Invalid column index (DBGrid Control) (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjDBGrid"}

This error results from setting a grid property or collection that requires a column index when the specified index is invalid.

## Invalid row number (DBGrid Control) (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjDBGrid"}

This error results from accessing the row property when the row is outside of the visible range of rows.



## Invalid selected row bookmark index (DBGrid Control) (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjDBGrid"}

An invalid index was specified for the SelBookmarks collection.

## Invalid setting for @0 Property (DBGrid Control) (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjDBGrid"}

An attempt was made to set a property to a value that is not valid for that property. Check the data type and range of the value.

## No current record (DBGrid Control) (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjDBGrid"}

Setting the grid's bookmark property and the grid can not move to the bookmark (ICursor move fails);  
or setting the editactive property and the grid can not move to a row (ICursor move fails).

## Null (DBGrid Control) (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjDBGrid"}

The "(null)" entry is used as the error text for some items that do not have an associated help context value for browser info. This will not normally appear as an actual error message.

## Operation is invalid within the event, @0 (DBGrid Control) (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjDBGrid"}

This error occurs when an illegal operation is attempted on the data control within the grid's notification events; for example, trying to refresh the data control in the AfterUpdate event.

## Property is not available in this context (DBGrid Control) (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjDBGrid"}

This error results from attempting to access a property that is available only within the context of the grid's Error event. For example, the ErrorText property is only readable from inside grid\_Error().

## Scroll arguments out of range (DBGrid Control) (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjDBGrid"}

This error results from setting the scroll property when the column index is out of range.

## Using the WithEvents keyword (Readme)

{ewc HLP95EN.DLL,DYNALINK,"See

Also":"rmhowUsingWithEventsKeywordC;vastmDim;vastmEvent;vastmPrivate;vastmPublic;vastmRaiseEvent"}

{ewc

HLP95EN.DLL,DYNALINK,"Example":"rmhowUsingWithEventsKeywordX;vastmEventX;vastmRaiseEventX"}

The **WithEvents** keyword is explained as part of certain declarative statements (**Dim**, **Public**, and **Private**). However, in order to use the statement, you need to understand the following two statements as well:

**Event** statement

**RaiseEvent** statement



## Coercion of Byte and String types (Readme)

In many previous versions of basic, strings were used to hold what were really arrays of byte data. Visual Basic now has a **Byte** data type, so this is unnecessary. Such strings can be assigned to resizable arrays of bytes. An array of bytes can similarly be assigned to a variable-length string. Be aware that coercion between types requires creation of temporary variables and arrays that may affect performance. Also, on Unicode platforms (all 32-bit operating systems), the same string contains twice as many bytes as on non-Unicode (16-bit operating systems) platforms.

## CreateToolWindow Method (Readme)

{ewc HLP95EN.DLL,DYNALINK,"Applies To":"rmmthCreateToolWindowMethodA;vaobjWindow"}

Creates a new Tool window containing the indicated DocObject.

### Syntax

*object*.**CreateToolWindow**(*AddInInst*, *ProgId*, *Caption*, *GuidPosition*, *DocObj*) **As Window**

The **CreateToolWindow** method syntax has these parts:

Part	Description
<i>object</i>	Required. An <u>object expression</u> that evaluates to an object in the Applies To list.
<i>AddInInst</i>	Required. <b>Addin</b> representing an addin in the development environment.
<i>ProgId</i>	Required. <b>String</b> representing the progID of the ActiveX document object.
<i>Caption</i>	Required. <b>String</b> containing the window caption
<i>GuidPosition</i>	Required. <b>String</b> containing a unique identifier for the window.
<i>DocObj</i>	Required. <b>Object</b> representing an ActiveX document object. This object will be set in the call to this function.

### Remarks

Member of **VBIDE.Windows**.

## DesignerID Property (Readme)

{ewc HLP95EN.DLL,DYNALINK,"Applies To":"rmproDesignerIDPropertyA;vaobjVBComponent"}

Read-only property that indicates the type of designer represented by the **VBComponent** object.

### Remarks

Member of **VBComponent**.

## FileName Method (Readme)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmthFileNameMethodC"} {ewc  
HLP95EN.DLL,DYNALINK,"Applies To":"rmmthFileNameMethodA;vaobjVBProject"} {ewc  
HLP95EN.DLL,DYNALINK,"Specifics":"rmmthFileNameMethodS"}
```

Returns a **String** containing the fully qualified path to the group project file.

### Syntax

*object*.**FileName()** As String

### Remarks

Member of **VBProject**.

## InsertFile Method (Readme)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmthInsertFileMethodC"} {ewc  
HLP95EN.DLL,DYNALINK,"Applies To":"rmmthInsertFileMethodA;vaobjVBComponent"} {ewc  
HLP95EN.DLL,DYNALINK,"Specifics":"rmmthInsertFileMethodS"}
```

Inserts code from a file into the code module.

### Syntax

*object*.**InsertFile**(*FileName* As String)

The *FileName* argument is a string representing the file containing the code to insert into the code module.

### Remarks

Member of **VBComponent**.

## LinkedWindows Property (Readme)

The **LinkedWindows** property is an accessor property (i.e., a property that returns an object of the same type as the property name).

## Properties Property (Readme)

The **Properties** property is an accessor property (i.e., a property that returns an object of the same type as the property name).

## References Property (Readme)

The **References** property is an accessor property (i.e., a property that returns an object of the same type as the property name).



## Executing VBA Addin Model Examples (Readme)

Use the following steps to execute example code for the VBA Addin model:

1. Start Visual Basic and choose Addin as the project type in the New Project dialog.
2. In the Project Explorer double click on the Connect class module to view its code.
3. Use Find from the Edit menu to search for OnConnection. This should place the cursor in the IDTExtensibility\_OnConnection procedure.
4. There is a comment 3 or 4 lines into the procedure suggesting the following statement is a good place to put a breakpoint for testing code. Place a breakpoint on the suggested line.
5. Place your cursor in the Immediate window, type AddToIni, then press Enter to execute that procedure. (AddToIni is a procedure in the module Addin.Bas.)
6. Press F5 to put the Addin in Run mode.
7. Start another instance of Visual Basic. Choose the default (Standard Exe) from the initial dialog. Then choose Add-In Manager from the Add-Ins menu.
8. Check My Addin-In on the list of Available Add-Ins. Press OK in the Add-In Manager dialog. The IDTExtensibility\_OnConnection is called in your first instance of Visual Basic execution is suspended at the breakpoint set in step 4.
9. Use Step Into from the debug menu to execute the line:  
    Debug.Print VBInst.FullName
10. You can now use VBInst as the object for the example code. Simply replace the dummy object Application.VBE with the Visual Basic object VBInst before executing the example lines in the Immediate window. For example, you can modify the example  
    Print Application.VBE.VBProjects(1).VBComponents.Count  
to read as follows:  
    Print VBInst.VBProjects(1).VBComponents.Count

When you press Enter on the latter line in the Immediate window the number of VB components is printed on the next line.

## Errors in callback procedures (Readme)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmhowErrorsInCallbackProceduresC"} {ewc  
HLP95EN.DLL,DYNALINK,"Applies To":"rmhowErrorsInCallbackProceduresA"} {ewc  
HLP95EN.DLL,DYNALINK,"Specifics":"rmhowErrorsInCallbackProceduresS"} {ewc
```

A callback procedure is a procedure within your program that Windows can call at certain times. You enable Windows to call this procedure by passing a reference to the procedure as an argument in an API call that is preceded by the **AddressOf** operator. Since the caller of a callback is not within your program, it is important that an error in the callback procedure not be propagated back to the caller. You can accomplish this by place the **On Error Resume Next** statement at the beginning of the callback procedure.

## Reserved Property and Method names (Readme)

The following names cannot be used as property or method names because they belong to the underlying IUnknown and IDispatch interfaces: **QueryInterface**, **AddRef**, **Release**, **GetTypeInfoCount**, **GetTypeInfo**, **GetIDsOfNames**, **Invoke**. Using these names causes a compilation error.

## Clarification for DateAdd function (Readme)

The format of the return value for **DateAdd** is determined by the Control Panel settings, not by the format that is passed in the date argument.

## Additional VBA (Language) Error messages (Readme)

The following error messages appear in some VBA helpfiles, but may be missing from others:

### Cannot handle events for the object specified

Events for this object could not be handled.

### Unexpected compiler error

This error occurs when a completely unanticipated error occurs during compilation.

### Invalid use of base class name

You cannot use the name of a base class by itself. This error has the following causes and solutions:

- You tried to use the name of a base class by itself without making clear that you were trying to access the base class' default member.  
Place the base-class name within parentheses to indicate you want to access the default member.
- You used the base-class name in an expression but the member you were trying to access was ambiguously specified.  
Use a disambiguator ( for example, an exclamation point) between the base-class name and the member you are interested in.
- You used the base-class name in a **Set** statement as though it contained a reference to the class.  
Use the base-class name to retrieve a reference for example, using GetObject.

### Invalid Event Name

Event procedure names are constructed by joining the object name to the event name with an underscore. This error has the following causes and solutions:

- You used an underscore as part of the event name.  
Remove the underscore from the event name.

### Event Not Found

An event specified in a **RaiseEvent** statement must correspond to a defined event. This error has the following causes and solutions:

- You specified a name in a **RaiseEvent** statement, but the event definition cannot be found.  
Make sure the event name is spelled correctly.

For additional information, select the item in question and press F1.

### Search string must be specified

You have to enter a string when searching in the object browser. This error has the following causes and solutions:

- You initiated a search in the Object Browser, but didn't specify text to search for.  
Enter a search string.

### Cannot display specified name because it is hidden

Some names exist in a type library, but are marked as hidden. This error has the following causes and solutions:

- You specified a name that is in the type library, but it is marked as hidden.  
You cannot normally view hidden type library members. Choose Show Hidden Members on the object browser context menu to make hidden members visible. You can then view the member information.

### **Cannot jump to specified type because it is in the specified library, which is not currently referenced**

This error has the following causes and solutions:

- You tried to specify a type in a library that isn't reference within the project.  
Set a reference to the type library through the References dialog.

### **Invalid inside Enum**

Not all types are valid within an enumeration definition. This error has the following causes and solutions:

- You tried to specify a string or some other invalid type as the value of an **Enum** member.  
The constant expression used to specify an **Enum** member must evaluate to type **Long** or another **Enum** type.

### **This component doesn't support the set of events (Error 459)**

Not every component supports client sinking of events. This error has the following cause and solution:

- You tried to use a **WithEvents** variable with a component that can't work as an event source for the specified set of events. For example, you may be sinking events of an object, then create another object that **Implements** the first object. Although you might think you could sink the events from the implemented object, that isn't automatically the case. **Implements** only implements an interface for methods and properties.  
You can't sink events for a component that doesn't source events.

## ClassName Property (ReadMe)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproClassNamePropertyC"}  
HLP95EN.DLL,DYNALINK,"Example":"vbproClassNamePropertyX":1}  
To":"vbobjvbcontrol;vbproClassNamePropertyA"} {ewc  
HLP95EN.DLL,DYNALINK,"Specifics":"vbproClassNamePropertyS"} {ewc  
{ewc HLP95EN.DLL,DYNALINK,"Applies
```

Returns the class name of a control.

### Syntax

*object*.**ClassName**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

## ContainedVBControls Property (ReadMe)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproContainedVBControlsPropertyC"} {ewc  
HLP95EN.DLL,DYNALINK,"Example":"vbproContainedVBControlsPropertyX":1} {ewc  
HLP95EN.DLL,DYNALINK,"Applies To":"vbcolcontainedvbcontrolscollection;vbproContainedVBControlsPropertyA"}  
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproContainedVBControlsPropertyS"}
```

Returns the collection of contained controls associated with a component.

### Syntax

*object*.**ContainedVBControls**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.



## VBControls Property (ReadMe)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcolVBControlsCollection;vbproVBControlsPropertyC"} {ewc  
HLP95EN.DLL,DYNALINK,"Example":"vbproVBControlsPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies  
To":"vbproVBControlsPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproVBControlsPropertyS"}
```

Returns a collection containing all controls on a form.

### Syntax

*object*.**VBControls**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

## rdoColumns Property (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproRdoColumnsPropertyC"} {ewc  
HLP95EN.DLL,DYNALINK,"Example":"vbproRdoColumnsPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies  
To":"vbproRdoColumnsPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRdoColumnsPropertyS"}

Contains stored **rdoColumn** objects.

### Syntax

*object*.**rdoColumns**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

## rdoConnections Property (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproRdoConnectionsPropertyC"} {ewc  
HLP95EN.DLL,DYNALINK,"Example":"vbproRdoConnectionsPropertyX":-1} {ewc HLP95EN.DLL,DYNALINK,"Applies  
To":"vbproRdoConnectionsPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRdoConnectionsPropertyS"}

Contains all open **rdoConnection** objects opened in an **rdoEnvironment** object.

### Syntax

*object*.**rdoConnections**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

## rdoEnvironments Property (ReadMe)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproRdoEnvironmentsPropertyC"}
HLP95EN.DLL,DYNALINK,"Example":"vbproRdoEnvironmentsPropertyX":1}
To":"vbproRdoEnvironmentsPropertyA"}
HLP95EN.DLL,DYNALINK,"Specifics":"vbproRdoEnvironmentsPropertyS"}
{ewc
{ewc HLP95EN.DLL,DYNALINK,"Applies
```

Contains all active **rdoEnvironment** objects of the **rdoEngine** object.

### Syntax

*object*.**rdoEnvironments**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

## rdoErrors Property (ReadMe)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproRdoErrorsPropertyC"} {ewc  
HLP95EN.DLL,DYNALINK,"Example":"vbproRdoErrorsPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies  
To":"vbproRdoErrorsPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRdoErrorsPropertyS"}
```

Contains all stored **rdoError** objects.

### Syntax

*object*.**rdoErrors**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

## rdoParameters Property (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproRdoParametersPropertyC"} {ewc  
HLP95EN.DLL,DYNALINK,"Example":"vbproRdoParametersPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies  
To":"vbproRdoParametersPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRdoParametersPropertyS"}

Contains all **rdoParameters** objects of an **rdoQuery**.

### Syntax

*object*.**rdoParameters**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

## rdoQueries Property (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproRdoQueriesPropertyC"} {ewc  
HLP95EN.DLL,DYNALINK,"Example":"vbproRdoQueriesPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies  
To":"vbproRdoQueriesPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRdoQueriesPropertyS"}

Contains all **rdoQuery** objects of an **rdoQuery**.

### Syntax

*object*.**rdoQueries**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

## rdoResultSets Property (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproRdoResultsSetsPropertyC"} {ewc  
HLP95EN.DLL,DYNALINK,"Example":"vbproRdoResultsSetsPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies  
To":"vbproRdoResultsSetsPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRdoResultsSetsPropertyS"}

Contains all open **rdoResultSet** objects in an **rdoConnection**.

### Syntax

*object*.**rdoResultSet**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.



## rdoTables Property (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproRdoTablesPropertyC"} {ewc  
HLP95EN.DLL,DYNALINK,"Example":"vbproRdoTablesPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies  
To":"vbproRdoTablesPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRdoTablesPropertyS"}

Contains all **rdoTable** objects in a database.

### Syntax

*object*.**rdoTables**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

## Changes to sample code in AddToAddInToolbar Method (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See

Also":"vbmscChangesToSampleCodeInAddToAddInToolbarMethodC;vbmtHaddtoaddintoolbarmethod"}

There is a comma missing from the `AddToAddInToolbar` line. It should instead be:

```
x.AddToAddInToolbar ("C:\VB5\MyAdd.DLL", _  
"MyAddIn.Connect", "MyAddIn Title", True, True)
```

## Changes to sample code in Design Considerations for Add-Ins (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmscChangesToSampleCodeInDesignConsiderationsForAddInsC"}

The complete path to access the TabOrder sample application is:

`\Vb\Samples\CompTool\AddIns\TabOrder.vbp.`

The complete path to access the Guidgen tool is: `\Tools\Idgen\Guidgen.exe.`

## Changes to sample code in Manipulating Code with Add-Ins (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See

Also": "vaobjCodeModule;vbmscChangesToSampleCodeInManipulatingCodeWithAddInsC"}  
}

For the included code samples, assume that the code line `Public vbi As VBIDE.VBE` is in the General Declarations section of the code.

## Changes to sample code in Manipulating the IDE with Add-Ins (ReadMe)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscChangesToSampleCodeInManipulatingIDEWithAddInsC"}
```

The **LoadPicture** statement is missing from the `Clipboard.SetData` line in the code sample. It should instead be:

```
Clipboard.SetData LoadPicture("C:\windows\circles.bmp")
```

## Correction in Creating a Basic Add-In (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscCorrectionInCreatingBasicAddInC"}

Step #4 of "Creating a Basic Add-In" erroneously states that the **Instancing** property of the add-in's Class module should be set to InSameProcess. It should instead be set to 5 - MultiUse.

## Changes to LinkedWindows Collection (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjLinkedWindows;vbcolChangesToLinkedWindowsCollectionReadmeC"}

In the last sentence of the section on the LinkedWindows collection, "...have a **ContainedWindows** collection" should be changed to "...have a **LinkedWindows** collection".

## UpdateCriteria constants erroneously listed in OpenResultset LockType "methods" list (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See

Also": "rdmthOpenResultset;rmmscUpdateCriteriaListedInOpenResultsetLockTypemethodsListreadmeC"}

When you use the RDO **OpenResultset** method and position the cursor over the *LockType* argument, and then use the right mouse button to select "List Properties and Methods", the LockType constants are listed, but two UpdateCriteria constants are erroneously listed as well (**rdCriteriaAllCols** and **rdCriteriaKey**).



## No rows updated or deleted (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgNoRowsUpdatedOrDeletedC"}

You get the error message "No rows updated or deleted" when using relative move methods (such as MovePrevious, MoveNext, and so on) with a dynamic cursor resultset.

This can occur when you use a dynamic cursor if another user has deleted rows just above or below your current rowset window (that is, the rows you have cached on your client). When you move toward the deleted rows, some of the rows in your current window will come back again since you are asking for the previous or next rowset size of rows. When you're near the end or beginning of a resultset, you can get data overlap since there's not enough rows to get a unique new set.

Setting a lower rowset size can help solve this problem. It degrades performance, but if performance is a concern, you're better off not using a dynamic cursor in the first place, since they're not the speediest performers when moving compared to other cursors.

Because of this, and because of the problem with error messages when data has been deleted, it is recommended that you avoid using dynamic cursors unless you need them and know what you're doing. A keyset or static cursor is much easier to deal with than a dynamic cursor, and a client-side batch static is probably the best for most client/server development.

## RDC Validate event help topic says that Actions can be changed (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscRDCValidateEventHelpTopicSaysThatActionsCanBeChangedC"}

The help topic for the RDC (**Remote Data Control**) Validate event states: "You can change the various *Move* methods and the **AddNew** method, which can be freely exchanged (any *Move* into **AddNew**, any *Move* into any other *Move*, or **AddNew** into any *Move*). Attempting to change **AddNew** or one of the *Moves* into any of the other actions is either ignored or produces a trappable error."

This is not possible. Changing the action to anything else results in the action being halted (the same as if you set it to **rdActionCancel**).

## Copy Method (Extensibility) (ReadMe)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthCopyMethodExtensibilityC"}  
HLP95EN.DLL,DYNALINK,"Example":"vbmthCopyMethodExtensibilityX":1}  
To":"vbcolselectedvbcontrols;vbmthCopyMethodExtensibilityA"} {ewc  
HLP95EN.DLL,DYNALINK,"Specifics":"vbmthCopyMethodExtensibilityS"}
```

```
{ewc  
{ewc HLP95EN.DLL,DYNALINK,"Applies
```

Copies the selected controls on the form to the **Clipboard**.

### Syntax

*object*.**Copy**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

## Cut Method (Extensibility) (ReadMe)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthCutMethodExtensibilityC"}  
HLP95EN.DLL,DYNALINK,"Example":"vbmthCutMethodExtensibilityX":1}  
To":"vbcolselectedvbcontrols;vbmthCutMethodExtensibilityA"} {ewc  
HLP95EN.DLL,DYNALINK,"Specifics":"vbmthCutMethodExtensibilityS"}
```

```
{ewc  
{ewc HLP95EN.DLL,DYNALINK,"Applies
```

Deletes the selected controls from the form.

### Syntax

*object*.**Cut**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

## AddIn Constants (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstAddInsConstantsC;vbidxVisualBasicConstants"}

### Connect Mode

Constant	Value	Description
<b>vbext_cm_AfterStartup</b>	0	Specifies the add-in's connection mode.
<b>vbext_cm_Startup</b>	1	Add-in is connected on IDE startup.
<b>vbext_cm_External</b>	2	Add-in is connected externally from Visual Basic.

### Disconnect Mode

Constant	Value	Description
<b>vbext_dm_HostShutdown</b>	0	Add-in is disconnected when the host shuts down.
<b>vbext_dm_UserClosed</b>	1	Add-in is disconnected by the user.

### Codepane View

Constant	Value	Description
<b>vbext_cv_ProcedureView</b>	0	Displays the code pane in Procedure view.
<b>vbext_cv_FullModuleView</b>	1	Displays the code pane in Full-module view.

### Component Types

Constant	Value	Description
<b>vbext_ct_StdModule</b>	1	The component is a standard module.
<b>vbext_ct_ClassModule</b>	2	The component is a class module.
<b>vbext_ct_MSForm</b>	3	The component is a form.
<b>vbext_ct_ResFile</b>	4	The component is a resource file.
<b>vbext_ct_VBForm</b>	5	The component is a Visual Basic form.
<b>vbext_ct_VBMDIForm</b>	6	The component is an MDI form.
<b>vbext_ct_PropPage</b>	7	The component is a property page.
<b>vbext_ct_UserControl</b>	8	The component is a <b>UserControl</b> object.
<b>vbext_ct_DocObject</b>	9	The component is a User document.
<b>vbext_ct_RelatedDocument</b>	10	The component is a RelatedDocument object.
<b>vbext_ct_ActiveXDesigner</b>	11	The component is a base class.

## Control Type

Constant	Value	Description
vbext_ct_Light	1	Light-weight control.
vbext_ct_Standard	2	Standard control.
vbext_ct_Container	3	Container control.

## File Types

Constant	Value	Description
vbext_ft_Form	0	The file is of type Form.
vbext_ft_Module	1	The file is of type Module.
vbext_ft_Class	2	The file is of type Class.
vbext_ft_Project	3	The file is of type Project.
vbext_ft_Exe	4	The file is of type .exe.
vbext_ft_Frx	5	The file is of type .frx.
vbext_ft_Res	6	The file is of type resource.
vbext_ft_UserControl	7	The file is of type UserControl.
vbext_ft_PropertyPage	8	The file is of type PropertyPage.
vbext_ft_DocObject	9	The file is of type DocObject.
vbext_ft_Binary	10	The file is of type Binary.
vbext_ft_GroupProject	11	The file is of type GroupProject.
vbext_ft_Designers	12	The file is of type Designer.

## Member Type

Constant	Value	Description
vbext_mt_Method	1	Member is of type Method.
vbext_mt_Property	2	Member is of type Property.
vbext_mt_Variable	3	Member is of type Variable.
vbext_mt_Event	4	Member is of type Event.
vbext_mt_Const	5	Member is of type Constant.

## Procedure Type

Constant	Value	Description
vbext_pk_Proc	0	Specifies all procedures other than property procedures.
vbext_pk_Let	1	Specifies a procedure that assigns a value to a property.
vbext_pk_Set	2	Specifies a procedure that sets a reference to an object.
vbext_pk_Get	3	Specifies a procedure that returns the value of a property.

## Project Start Mode

Constant	Value	Description
vbext_psm_StandAlone	0	Startup mode is stand alone.

<b>vbext_psm_OleServer</b>	1	Startup mode is ActiveX component.
----------------------------	---	------------------------------------

### Project Type

Constant	Value	Description
<b>vbext_pt_StandardExe</b>	0	Project is a standard .exe project.
<b>vbext_pt_ActiveXExe</b>	1	Project is an ActiveX .exe project.
<b>vbext_pt_ActiveXDll</b>	2	Project is an ActiveX DLL project.
<b>vbext_pt_ActiveXControl</b>	3	Project is an ActiveX control project.

### Run (VBA) Mode

Constant	Value	Description
<b>vbext_vm_Run</b>	0	The project is in run mode.
<b>vbext_vm_Break</b>	1	The project is in break mode.
<b>vbext_vm_Design</b>	2	The project is in design mode.

### Reference Type

Constant	Value	Description
<b>vbext_rk_TypeLib</b>	0	Represents a reference to a type library.
<b>vbext_rk_Project</b>	1	Represents a reference to a project.

### Scope

Constant	Value	Description
<b>vbext_Private</b>	1	Member has private scope.
<b>vbext_Public</b>	2	Member has public scope.
<b>vbext_Friend</b>	3	Member has friend scope.

### Startup Object

Constant	Value	Description
<b>vbext_so_SubMain</b>	0	Returns a variant containing the startup component for the project.
<b>vbext_rk_Project</b>	1	Returns a variant containing the startup component for the project.

### Display Mode

Constant	Value	Description
<b>vbext_dm_SDI</b>	0	The display mode is single document interface.
<b>vbext_dm_MDI</b>	1	The display mode is multiple

document interface.

### Window State

Constant	Value	Description
<b>vbext_ws_Normal</b>	0	Normal window state.
<b>vbext_ws_Minimize</b>	1	Window is minimized to an icon.
<b>vbext_ws_Maximize</b>	2	Window is maximized (enlarged to its maximum state).

### Window Types

Constant	Value	Description
<b>vbext_wt_CodeWindow</b>	0	The window is a code window.
<b>vbext_wt_Designer</b>	1	The window is a designer window.
<b>vbext_wt_Browser</b>	2	The window is a Browser window
<b>vbext_wt_Watch</b>	3	The window is a watch window.
<b>vbext_wt_Locals</b>	4	The window is a locals window.
<b>vbext_wt_Immediate</b>	5	The window is an Immediate window.
<b>vbext_wt_ProjectWindow</b>	6	The window is a Project window.
<b>vbext_wt_PropertyWindow</b>	7	The window is a Property window.
<b>vbext_wt_Find</b>	8	The window is a Search window.
<b>vbext_wt_FindReplace</b>	9	The window is a Find and Replace window.
<b>vbext_wt_Toolbox</b>	10	The window is a Toolbox window.
<b>vbext_wt_LinkedWindowFrame</b>	11	The window is a linked window frame.
<b>vbext_wt_MainWindow</b>	12	The window is a main window.
<b>vbext_wt_Preview</b>	13	The window is a preview window.
<b>vbext_wt_ColorPalette</b>	14	The window is a color palette window.
<b>vbext_wt_ToolWindow</b>	15	The window is a Tool window.

### Menu Shortcuts

Constant	Value	Description
<b>vbextMenuShortcutCtrlA</b>	1	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlB</b>	2	User-defined shortcut



		keystroke.
<b>vbextMenuShortcutCtrlI</b>	3	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlID</b>	4	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIE</b>	5	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIF</b>	6	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIG</b>	7	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIH</b>	8	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlII</b>	9	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIJ</b>	10	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIK</b>	11	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIL</b>	12	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIM</b>	13	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIN</b>	14	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIO</b>	15	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIP</b>	16	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIQ</b>	17	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIR</b>	18	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIS</b>	19	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIT</b>	20	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIU</b>	21	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIV</b>	22	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIW</b>	23	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIX</b>	24	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIY</b>	25	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIZ</b>	26	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIF1</b>	39	User-defined shortcut

		keystroke.
<b>vbextMenuShortcutCtrlF2</b>	40	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlF3</b>	41	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlF4</b>	42	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlF5</b>	43	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlF6</b>	44	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlF7</b>	45	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlF8</b>	46	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlF9</b>	47	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlF10</b>	48	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlF11</b>	49	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlF12</b>	50	User-defined shortcut keystroke.
<b>vbextMenuShortcutShiftF1</b>	51	User-defined shortcut keystroke.
<b>vbextMenuShortcutShiftF2</b>	52	User-defined shortcut keystroke.
<b>vbextMenuShortcutShiftF3</b>	53	User-defined shortcut keystroke.
<b>vbextMenuShortcutShiftF4</b>	54	User-defined shortcut keystroke.
<b>vbextMenuShortcutShiftF5</b>	55	User-defined shortcut keystroke.
<b>vbextMenuShortcutShiftF6</b>	56	User-defined shortcut keystroke.
<b>vbextMenuShortcutShiftF7</b>	57	User-defined shortcut keystroke.
<b>vbextMenuShortcutShiftF8</b>	58	User-defined shortcut keystroke.
<b>vbextMenuShortcutShiftF9</b>	59	User-defined shortcut keystroke.
<b>vbextMenuShortcutShiftF10</b>	60	User-defined shortcut keystroke.
<b>vbextMenuShortcutShiftF11</b>	61	User-defined shortcut keystroke.
<b>vbextMenuShortcutShiftF12</b>	62	User-defined shortcut keystroke.
<b>vbextMenuShortcutShiftDel</b>	78	User-defined shortcut keystroke.
<b>vbextMenuShortcutShiftIns</b>	76	User-defined shortcut

		keystroke.
<b>vbextMenuShortcutDel</b>	77	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlIns</b>	75	User-defined shortcut keystroke.
<b>vbextMenuShortcutAltBksp</b>	79	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlShiftF1</b>	51	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlShiftF2</b>	52	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlShiftF3</b>	53	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlShiftF4</b>	54	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlShiftF5</b>	55	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlShiftF6</b>	56	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlShiftF7</b>	57	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlShiftF8</b>	58	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlShiftF9</b>	59	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlShiftF10</b>	60	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlShiftF11</b>	61	User-defined shortcut keystroke.
<b>vbextMenuShortcutCtrlShiftF12</b>	62	User-defined shortcut keystroke.

### Source Code Control

<b>Constant</b>	<b>Value</b>	<b>Description</b>
<b>vbextSCCStatusNotControlled</b>	0	File is not under source code control.
<b>vbextSCCStatusControlled</b>	1	File is under source code control.
<b>vbextSCCStatusCheckedOut</b>	2	File is checked out to the current user.
<b>vbextSCCStatusOutOther</b>	4	File is checked out to another user.
<b>vbextSCCStatusOutOfDate</b>	32	File is not the most recent.
<b>vbextSCCStatusShared</b>	512	File is shared between projects.

## Avoid overlapping of dynamic cursors (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See

Also": "daobjDynamicTypeRecordset;rmmscAvoidingOverlappingOfDynamicCurorsReadMeC"}

If you use a dynamic cursor, and another user has deleted rows just above or below your current rowset window (that is, the rows you have cached on your client), you can get data error messages. This occurs because when you move toward the deleted rows, some of the rows in your current window will come back again, since you are asking for the previous or next rowset size of rows. When you're near the end or beginning of a resultset, you can get data overlap since there's not enough rows to get a unique new set.

Setting a lower rowset size can help solve this problem. It degrades performance, but if performance is a concern, you're better off not using a dynamic cursor in the first place, since they're not the speediest performers when moving compared to other cursors.

Because of this, and because of the problem with error messages when data has been deleted, it is recommended that you avoid using dynamic cursors unless you need them and know what you're doing. A keyset or static cursor is much easier to deal with than a dynamic cursor, and a client-side batch static is probably the best for most client/server development.

## Bug with Testing Procedures with the Immediate Window topic (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscBugWithTestingProceduresWithImmediateWindowTopicC"}

In Chapter 13 of the Programmer's Guide in the topic "Testing Procedures with the Immediate Window" the documentation incorrectly states: "If Option Explicit is in effect... any variables you enter in the Immediate window must already be declared within the current scope." The Option Explicit statement has no effect on variables entered in the Immediate window. Note however that any variables entered in the Immediate window that have not been previously declared as a specific type will be treated as Variants.

## Compile error in Form Unload sample code (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscCompileErrorInSampleAppC"}

In Chapter 6 of the Programmer's Guide in the topic "More About Forms" the sample code is incorrect and will cause a compile error if copied into a project.

The incorrect code is:

```
Private Sub Form_Unload()  
    Dim i As Integer  
    'Loop through the forms collection and unload  
    'each form.  
    For i = 0 To Forms.Count - 1  
        Unload Forms(i)  
    Next  
End Sub
```

In this example, the "Cancel as Integer" parameter is missing from the Form\_Unload declaration; also the forms must be unloaded in reverse order to avoid a "subscript out of range" error.

The correct code is:

```
Private Sub Form_Unload (Cancel As Integer)  
    Dim i As Integer  
    'Loop through the forms collection and unload  
    'each form.  
    For i = Forms.Count - 1 To 0 Step -1  
        Unload Forms(i)  
    Next  
End Sub
```

## Missing Documentation: ValueTips and Bookmarks (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscMissingValueTipsC"}

The documentation fails to mention two new features in VB5: ValueTips in the Immediate window, and Bookmarks in the Code Editor.

ValueTips are similar to ToolTips except that they display the current value when the mouse is held over a variable or object property in the Code Window in Break mode. The display of ValueTips is limited to variables and objects that are currently in scope.

ValueTips are also available in the Immediate window in Break mode. Unlike the Code Editor, the Immediate window will display values for object properties regardless of scope if a fully qualified object name is provided. For example, a ValueTip would always be displayed for Form1.Text1.Width, but not for Text1.Width unless Text1 was currently in scope.

Bookmarks are a new feature of the Code Editor that can be used to mark lines of code so that you can easily return to them later. Commands to toggle bookmarks on or off as well as to navigate existing bookmarks are available from the Edit, Bookmarks menu item, or from the Edit toolbar

## Printing Books Online (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscVB5BugPrintingBooksOnlineC"}

You can print topics from Books Online by using Print on the File menu (to print individual topics) or by adding topics to the Notebook (to print multiple topics). Topics printed using the notebook do not include the topic titles.

An easier way to print multiple topics (including their titles) is to use the Word Viewer and Word documents provided in the Tools directory on the Visual Basic CD-ROM.



## Palette Property - Additional Information (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscPalettePropertyAdditionalInformationReadMeC"}

The Palette Property topic in online Help states "You can use a .dib, .gif, or .pal file to set the palette as well as .bmp files." This is incorrect. Only .dib, .gif, and .bmp files can be assigned to the Palette property. Assigning a .pal file will cause a run-time error 481: "Invalid picture".

## Using Relative Palette Colors (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"Applies To":"rmmscUsingRelativePaletteColorsA"}

When designing for 256 color displays, some colors may appear dithered. This can make text and other graphic elements difficult to read. By specifying a relative palette color, Visual Basic will display the closest undithered approximation of a specified color on 256 color displays while still displaying the exact color at higher color depths.

To force Visual Basic to use the closest solid, rather than dithered, color for a given property, put a 2 in the high order byte of the color property. For example, to force a form's background to be a solid light orange you could use the following code:

```
Private Function PaletteRGB( RGB As Long) As Long
    PaletteRGB = &H02000000 Or RGB
End Function
```

If you set this at design time:

```
Form1.BackColor = &H00C0E0FF&      'dithered light orange
```

And add the following to the Form\_Click event:

```
Private Sub Form_Click()
    Form1.BackColor = PaletteRGB( Form1.BackColor)
End Sub
```

At run-time when the form is clicked the backcolor will change to a solid, rather than dithered, shade. It is now using the closest color out of the halftone palette. This effect may not be visible on systems running at a color depth greater than 256 colors.

## Programming with Objects - Error in Code Sample (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"Applies To":"rmmscProgramminigWithObjectErrorInCodeSampleReadMeA"}

In the topic "Putting Property Procedures to Work for You" in "Programming with Objects" there is an error in the code example demonstrating that the arguments of paired property procedures must match.

It states:

The Property Get declaration must use arguments with the same name and data type as the arguments in the Property Let Procedure.

```
Public Property Let Things(ByVal X As Integer...
```

The "Let" in this line of code should be replaced with "Get".

## Setting Combo Box Styles at Run Time (ReadMe)

Combo Box styles are read-only and can only be set at design time. Chapter 7 of the Programmer's Guide incorrectly states that Combo Box styles can be set at both design and run time. If you attempt to set the style of a Combo Box at run time, you will receive an error message.

## Setup Wizard Looking for "User" or "Kernel" (ReadMe)

If you create an application that includes conditional compilation (for example, "#If Win 16 Then") and 16 bit declares, the Setup Wizard will query you with "Locate KERNEL" or "Locate USER." This occurs when you use the following declaration:

```
Declare Function GetVersion16 Lib "Kernel" Alias "GetVersion" () As Long
```

To avoid this situation, change the declaration from "Kernel" to "Kernel.dll." This results in the line below:

```
Declare Function GetVersion16 Lib "Kernel.dll" Alias "GetVersion" () As Long
```

## Hide Method Fails When Repeated in Modal Forms (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscHideMethodFailsWhenRepeatedInModalFormsC;vbmthHide"}

When closing a modal form that has been opened from another modal form, the following code worked in previous versions of Visual Basic:

```
Me.Hide  
Me.Hide ' This will now cause an error.
```

In Visual Basic 5.0 this code will now fail on the second "Me.Hide." You can substitute "Me.Hide" with "Me.Visible = False," as shown below:

```
Me.Visible = False  
Me.Visible = False ' No error raised.
```

## Application Performance Explorer Limited to 32 Profiles (Readme)

The built version of the Application Performance Explorer (APE) has a limit of approximately 32 profiles when running in Windows 95. The limitation is due to an apparent .INI file size limit in Windows 95. The profiles are stored in an .INI file that can be specified with the Set Profile Collection command under the File menu. You can determine how many profiles are contained in the present collection by clicking the Profile combo box. The dropdown list contains all profiles in the present collection.

To avoid losing data, when the collection nears 32, begin a new Profiles Collection.

### **To create a new Profiles Collection**

- 1** On the File menu, click Set Profile Collection.
- 2** In the dialog box named Choose Profile Collection, click the File name box and type in the name of a new Profile Collection.
- 3** Click Save.

## Internet Transfer Control: Set URL Before Password and UserName (ReadMe)

```
{ewc HLP95EN.DLL,DYNALINK,"See  
Also":"rmmsclInternetTransferControlSetURLBeforePasswordUserNameC;vbobjInternetControl"}
```

When using the **Internet Transfer** control, set the **URL** property before you set the **Password** and **UserName** properties. Currently, if you set the **URL** property last, the **UserName** and **Password** properties will be set to "". This code will work:

```
Inet1.URL = "FTP://ftp.myCompany.com"  
Inet1.Password = "I(3Lei#4"  
Inet1.UserName = "Jonne Smythe"  
Text1.Text = Inet1.OpenURL
```

The code below, however, will fail:

```
Inet1.Password = "I(3Lei#4"  
Inet1.UserName = "Jonne Smythe"  
Inet1.URL = "FTP://ftp.myCompany.com" ' PassWord and  
                                         ' Username wiped out.  
Text1.Text = Inet1.OpenURL
```



## Don't Share a Property Page Between Projects if a UserControl Is on It (ReadMe)

Although it is possible to place a **UserControl** on a property page, and to share that property page between projects (within the same Project group), you will not be able to update the control on the property page. For this reason, you should avoid sharing a property page that has a **UserControl** object on it. The specific scenario where this happens is shown below:

### To share a Property Page with a UserControl on it

- 1 On the File menu, click New Project.
- 2 Double-click the ActiveX Control icon.
- 3 On the Project menu, click Add Property Page.
- 4 Double-click the Property Page icon to add it to the project.
- 5 Repeat steps 2-4 to add a second control project and property page to the project group.
- 6 On the File menu, click Save Project Group. When prompted to overwrite the control project and property page, click OK.
- 7 Close all of the designers in the project group except for Project2's PropertyPage1 designer.
- 8 On the Toolbox, double-click UserControl1 to add it to the PropertyPage.

If you attempt to close the property page designer, you will get an error. Further efforts to modify the UserControl object will also fail.

## ListView Tooltips Cause ZOrder Problems When Used in Addins (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See

Also": "rmmscListViewToolTipsCauseZOrderProblemsWhenUsedInAddinsC;vbobjListView"}

When using the **ListView** control as part of an Addin, some problems with zorder may be caused by Tooltips. The following conditions must all be true in order to manifest these problems:

- The **ListView** control is used in an Addin.
- The **ListView** control's **View** property is set to Large Icons.
- The Tooltip is long enough to wrap.

When these problems occur, they can only be remedied by shutting down Visual Basic.

## Access95 Won't Run After Uninstalling VB4 (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscAccess95WontRunAfterUninstallingVB4C"}

If you uninstall Visual Basic version 4 after installing Access 95, Access will no longer work. This problem will not occur with Visual Basic version 5. To get Access running again, re-register the old version of DAO (DAO3032.DLL).

Use REGSVR32.EXE to re-register DAO. REGSVR32.EXE is usually found in the \tools\regutils folder, and DAO3032.DLL is usually in the \program files\common files\microsoft shared\dao folder. The command line to re-register DAO is:

```
cd "c:\program files\common files\microsoft shared\dao"  
regsvr32 dao3032.dll
```

## Microsoft Repository Installation (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscMicrosoftRepositoryInstallationReadmeC"}

To install Microsoft Repository 1.0, during the Visual Basic Enterprise Edition installation, you must choose "Custom" installation, and check the box for Microsoft Repository. By default, this box is unchecked.

Microsoft Repository requires that the Data Access component is also installed. By default, the box for the Data Access component is checked. However, if you uncheck this box and install Microsoft Repository without the Data Access component, you will encounter this error message when you start Visual Basic 5.0:

"An error occurred while opening the Microsoft Repository database. Specified driver could not be loaded due to system error 126 (Microsoft Access Driver (\*.MDB)). Microsoft Repository Add-in for Visual Basic is shutting down."

To correct this problem, install the Data Access component.

## Obsolete Files from Visual Basic Beta (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscObsoleteFilesFromVisualBasicBetaReadmeC"}

If you install this version of Visual Basic 5.0 over a previous installation of a Visual Basic 5.0 beta release, there are a few Microsoft Repository files that are obsolete and can be deleted to save space.

Starting from the directory to which you install Visual Basic 5.0, look for and delete these files:

- REPOSTRY\README.WRI
- REPOSTRY\BIN\BROWSER.EXE

In your Windows system directory, look for and delete these files:

- REPAUTO.CNT
- REPAUTO.HLP
- REPCOM.CNT
- REPCOM.HLP

# Repository API Changes (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscRepositoryAPIChangesReadmeC"}

- **TXN\_QUERY\_TIMEOUT** is a new type of option that can be set and retrieved via the **SetOption** and **GetOption** methods of the **IRepositoryTransaction** interface. The value for this option specifies the number of seconds that an outstanding query is allowed to execute before it is canceled. The default value for the option is 120 seconds.
- **TXN\_DBMS\_READONLY** is a new type of option that can be retrieved via the **GetOption** method of the **IRepositoryTransaction** interface. The value for this option specifies whether or not you can make changes to the currently open repository database. If the value of the option is zero, you can make changes to the database. If the value is nonzero, you cannot make changes to the database.
- The maximum length of an annotational property is 220 characters, not 255 characters. The constant **PROPVALSIZE** determines this limit.
- The **Interface** method of the **IRepositoryItem** interface is used to obtain an interface pointer to an alternate interface that an Automation object exposes. This method will accept one of three different types of input parameter to specify the interface whose pointer is to be retrieved: the object identifier of the interface definition object (a repository object identifier), the interface identifier for the interface (a GUID), or the name of the interface (a character string).
- A new method (**ExecuteQuery**) has been added to the **IRepositoryODBC** interface. The **ExecuteQuery** method has this signature:

HRESULT ExecuteQuery(BSTR QueryString, IObjectCol \*\*pplCol)

QueryString is a character string that contains an SQL query (or the name of a stored procedure) that is to be executed against the open repository database. The selected columns that are returned by the SQL query must consist of either the internal identifier column (IntID), or the internal identifier column and the type identifier column (TypeID). The results of the query are returned as an object collection, and an **IObjectCol** interface pointer is returned to the caller via the \*pplCol output parameter.

- The **get\_Item** method of the **IObjectCol** interface does not support retrieving objects by name. The **get\_Item** method of the **ITargetObjectCol** and **IRelationshipCol** interfaces supports retrieving objects by name, but only when the object collection is a naming collection. The name is not required to be unique.
- The name assigned to a project via the naming relationship of the **Contents** collection of the **IMpoProjectItemContainer** interface is adorned to ensure that the name is unique among the collection of all Visual Basic projects in the collection. The adornment follows these rules:
  - If the project file is a network file, then the UNC name for the file is used as the name.
  - If the project file is a local file, then the string [computerName]filename is used as the name.
  - If the project is unsaved, then the string [computerName]~processId~projectName is used as the name.

The **FileName** property of the **IMpoProjectItem** interface contains the unadorned file name for a project.

- The **IReposProperties** interface is a dependent interface that is used to access the collection of repository properties that are attached to any interface that inherits from **IRepositoryDispatch**. As a dependent interface, it is only accessible by obtaining an interface pointer via the **get\_Properties** method of the **IRepositoryDispatch** interface. A new method (**get\_Type**) is supported on the **IReposProperties** interface. The **get\_Type** method has this signature:

HRESULT get\_Type(VARIANT \*psTypeID)

The **get\_Type** method retrieves the type of the interface that derives its base behavior from **IRepositoryDispatch**; that is, it returns the object identifier for the interface definition object to which the derived interface conforms.

## Repository Browser (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscRepositoryBrowserReadmeC"}

A basic browser is supplied with Microsoft Repository that enables you to browse the contents of a repository database. It is named REPBROWS.EXE, and can be found in the REPOSTRY\BIN directory immediately underneath the directory to which you install Visual Basic 5.0.

REPBROWS.EXE accepts four optional command line input arguments. They are:

**Access**=*filename*;  
**DSN**=*dataSourceName*;  
**UID**=*userID*;  
**PWD**=*password*;

Fill in the appropriate values for the italicized items. The semicolon is required on the end of each argument.

## Repository Database Restrictions (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscRepositoryDatabaseRestrictionsReadmeC"}

Microsoft Repository 1.0 can use either Jet or SQL Server to manage the repository database. These restrictions apply:

- If SQL Server is used, the version must be 6.5 or later.

If Jet is used, care must be taken to avoid large amounts of data being accessed within the scope of a single transaction. The Jet engine uses an in-memory cache to speed up query processing. This cache continues to grow inside of a transaction until a Commit or Abort is performed. If your repository application is accessing a large amount of data within a single transaction (which might be done to isolate the view of the retrieved data from uncommitted changes of other processes), the cache can grow to the point where it consumes enough memory to cause the application to fail. To avoid this situation, periodically Commit your transaction, even if it is a read-only transaction.



## Setup Fails On NT 4.0 With FastFind Installed (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscSetupFailsOnNT40WithFastFindInstalledReadmeC"}

When installing Visual Basic on a machine running Windows NT 4.0 and the FastFind feature of Microsoft Office 97, setup may be unable to register Msjet35.dll. This will cause setup to fail.

Temporarily disabling or removing FastFind may allow setup to succeed.

## ShowTips Property is Read-Only (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscShowTipsPropertyIsReadOnlyReadMeC;vbproTooltipsS"}

The correct syntax for this property is

*object*.**ShowTips**

Syntax in the Help topic incorrectly indicates the value can be set at run-time.

## Remove MV141KN.OCX From Your System Before Installing (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscRemoveMV141KNOCXFromYourSystemBeforeInstallingReadMeC"}

If you were a beta user of VB5, remove MV141KN.OCX from your system before installing the released product.

## Can't Pass a QueryDef Object Name to an OpenRecordset Method Using ODBCDirect (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See

Also": "rmmscCantPassQueryDefObjectNameToOpenRecordsetMethodUsingODBCDirectReadMeC"}

When you use ODBCDirect, you cannot open a **Recordset** object from a QueryDef method object name. Instead, you should perform the **OpenRecordset** method directly from the **QueryDef** object. This is because QueryDef objects in ODBCDirect are not permanent objects as they are in a Microsoft Jet database.

## Can't read BatchCollisionCount property through Recordset properties collection (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See

Also": "rmmscCantReadBatchCollisionCountPropertyThroughRecordsetPropertiesCollectionReadMeC"}

The Properties collection of a recordset lists a property named "BatchCollisionsCount". The property is actually named "BatchCollisionCount". If you try to read that property through the Properties collection, you will get an error "Property not found". The only way to read this property is directly, i.e. `MyRS.BatchCollisionCount`.

## dbFailOnError No Longer Rolls Back a Transaction (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscDbFailOnErrorNoLongerRollsBackTransactionReadMeC"}

In previous releases of Data Access Objects (DAO), if you executed SQL statements, they were internally treated as transactions. If you executed a statement with the dbFailOnError flag on and the query failed, the operation was rolled back. For performance reasons, an SQL statement is no longer treated as a transaction. Therefore, if an SQL query fails in Microsoft Access, an incomplete operation may occur. If you think an error may occur, you should explicitly use the statement within a transaction by using the BeginTrans method and the CommitTrans method. However, note that explicit transactions may slow query performance.

## ODBCDirect: GetRows Error 40035 (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscODBCDirectGetRowsError40035ReadMeC"}

You should not use the GetRows method with long value fields. If you use the GetRows method on an ODBCDirect Recordset object containing long value fields (Memo or Long Binary), you will get an error variant stored in the array wherever the long value field should have been. If you read the array data value, it will be Error 40035 and the data type will be Variant.

## RegisterDatabase truncates server to 31 characters (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscRegisterDatabaseTruncatesServerTo31CharactersReadMeC"}

When calling RegisterDatabase, the server parameter will be truncated to 31 characters if it exceeds that length. The workaround is to rename the server in the registry key to a shorter value.



## Using DAO Version 3.5 in Older OLE Automation Host Applications (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscUsingDAOVersion35InOlderOLEAutomationHostApplicationsReadMeC"}

Data Access Objects (DAO) version 3.5 is designed to work with Visual Basic for Applications version 5.0. Although DAO 3.5 will appear in Office 95 or Visual Basic version 4.0 or earlier reference lists, you can't use DAO 3.5 in either Microsoft Office 95 or Visual Basic version 4.0 or earlier.

## Repository Multithreading Restrictions (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscRepositoryMultithreadingRestrictionsReadMeC"}

When programming a multithreaded repository application, take care to synchronize repository database commit operations between your application threads. Specifically, you must synchronize the use of these C++ methods:

- IRelationshipCol::Count
- IClassDef::ObjectInstances
- IInterfaceDef::ObjectInstances
- IRepositoryODBC::ExecuteQuery

## If You Used Microsoft Repository from the Visual Basic Beta (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmsclIfYouUsedMicrosoftRepositoryFromVisualBasicBetaReadMeC"}

- If you install this version of Visual Basic 5.0 over a previous installation of a Visual Basic 5.0 beta release, there are a few Microsoft Repository files that are obsolete and can be deleted to save space.

Starting from the directory to which you install Visual Basic 5.0, look for and delete these files:

REPOSTRY\README.WRI

REPOSTRY\BIN\BROWSER.EXE

- In your Windows system directory, look for and delete these files:

REPAUTO.CNT

REPAUTO.HLP

REPCOM.CNT

REPCOM.HLP

- The beta format of the repository database is no longer valid. If you used the beta version of Microsoft Repository, you must delete your old repository database. In the MSAPPS\REPOSTRY directory underneath your Windows directory, look for and delete this file:

REPOSTRY.MDB

If you delete this file before you install the release version of Visual Basic 5.0 and you choose to install Microsoft Repository, the installation process will recreate this file. If you delete this file after installation and you chose to install Microsoft Repository, the first time that you start Visual Basic 5.0 the file will be recreated. In this case, you may notice that Visual Basic takes a little longer to start.

## Assigning Resultset to RDC Doesn't Update Bound Controls (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See  
Also":"rdproResultset;rmmsscAssigningResultsetToRemoteDataControlDoesNotUpdateControlsReadMeC"}

When you bind a control to the resultset of a RemoteData Control (RDC), the resultset doesn't automatically display in the control. To illustrate this:

- 1 Start Visual Basic and open a Standard EXE project.
- 2 Reference the RDC.
- 3 Place an RDC on the form.
- 4 Place a TextBox control on the form.
- 5 Set the following TextBox properties:

**DataSource:** MSRDC1

**DataField:** au\_lname

- 6 Place a CommandButton control on the form and add the following code to its Click event:

```
Dim cn As New rdoConnection
cn.Connect = _
    "dsn=pinkpearl;database=rdobugs;uid=rdo;pwd="
cn.EstablishConnection
Set MSRDC1.Resultset = cn.OpenResultset("select * _
    from authors]")
```

- 7 Run the project (F5).
- 8 Click the CommandButton.

Notice that the bound control does not populate with data. You must issue the command `MSRDC1.Refresh` for the bound control to populate. The refresh causes the server to send the entire resultset again, which can take a long time in some situations.

### Workaround:

To work around this problem, set any bound control's datafield after setting the resultset in code. For example, after the line

```
Set MSRDC1.Resultset = cn.OpenResultset("select * _
    from authors]")
```

you would add

```
Text1.DataField = "au_lname"
```

which forces the binding manager to set and update the bindings, which populates the bound control with data.

## Can't Assign Resultset to RemoteData Control (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See

Also":"rdidxRemoteDataObjectModel;rmmsscCantAssignResultsetToRemoteDataControlC"}

When you attempt to assign a forward-only resultset to a RemoteData Control (RDC), you get an "invalid object" error. To illustrate this situation:

- 1 Start Visual Basic.
- 2 Place a RemoteData Control on Form1.
- 3 Add a reference to RDO through the References command in the Project menu.
- 4 Add the following code to the Form\_Load event:

```
Dim x as new rdoConnection
Dim y as rdoQuery
x.Connect = "DSN=Union;UID=rdo;PWD="
x.EstablishConnection
Set y = x.CreateQuery("Query1", "SELECT * FROM _
authors")
x.Query1
' invalid object error occurs on next line
Set MSRDC1.Resultset = x.LastQueryResults
```

- 5 Press F5.

The reason this error occurs is that it uses a forward-only resultset which cannot be assigned to the RDC. In order to assign a resultset to an RDC, it must be either keyset or static. For example:

```
Dim x As New rdoConnection
Dim y As rdoQuery
x.Connect = "DSN=Union;database=rdobugs;UID=rdo;PWD="
x.EstablishConnection
Set y = x.CreateQuery("Query1", "SELECT * FROM _
authors")
y.CursorType = rdOpenKeyset
y.LockType = rdConcurRowVer
x.Query1
Set MSRDC1.Resultset = x.LastQueryResults
```

## GPF When Toggling Folders if UserDocument Object and Designers are Present (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscGPFWhenTogglingFoldersIfUserDocDesignersArePresentC"}

If you create an ActiveX EXE project and insert a **UserDocument** object and then an **ActiveX** designer, and then toggle the folders in the Project window from one state to another, the reference to the **UserDocument** disappears from the Project window.

In many cases, this can lead to a General Protection Fault (GPF), because Visual Basic searches through the Project window in order to save project items. If you toggle the folders and then attempt to save your project or shut down Visual Basic, and a project item that Visual Basic expects to be present in the project window disappears from the window—such as a **UserDocument** object—you will receive a GPF.

# IStudio/ActiveX ScriptPad Can't Change Fonts Using VB-created Font Property (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmsclStudioActiveXScriptPadHasProblemSavingFontsC"}

If you create a **Font** property in Visual Basic, then attempt to use it to change a font using **ScriptPad** or **IStudio**, the font is not changed. To illustrate the problem:

- 1 Start Visual Basic and create a new ActiveX Control project.
- 2 Use the Control Interface wizard to create a **Font** property. (Map it to `UserControl.Font`).
- 3 Add `On Error Resume Next` to the `WriteProperties` event.
- 4 Make sure that the `ProcID` for the **Font** property is set to "Font".
- 5 Build the ActiveX control.
- 6 Start ScriptPad or IStudio.
- 7 Place the control on the HTML page.
- 8 Change the font shown in the property sheet.
- 9 Close the designer.

Notice that the font is not changed.

This problem occurs because the `WriteProperties` event:

```
Call PropBag.WriteProperty("Font", UserControl.Font, _
    Ambient.Font)
```

never writes any font information because the **Font** object is equivalent to `Ambient.Font`, as demonstrated by the following code:

```
MsgBox IIF(Font IS Ambient.Font, "The Font object is _
    identical to Ambient.Font", "Different Objects")
```

The following code illustrates the problem:

```
'MappingInfo=UserControl,UserControl,-1,Font
Public Property Get Font() As Font
    Set Font = UserControl.Font
End Property
Public Property Set Font(ByVal New_Font As Font)
    Set UserControl.Font = New_Font
    PropertyChanged "Font"
End Property
'Initialize Properties for User Control
Private Sub UserControl_InitProperties()
    Set Font = Ambient.Font
End Sub
'Load property values from storage
Private Sub UserControl_ReadProperties(PropBag As _
    PropertyBag)
    Set Font = PropBag.ReadProperty("Font", _
    Ambient.Font)
End Sub
'Write property values to storage
Private Sub UserControl_WriteProperties(PropBag As _
    PropertyBag)
    Call PropBag.WriteProperty("Font", Font, _
    Ambient.Font)
End Sub
```

`UserControl_InitProperties` sets the font to be the ambient font. When you set the property in the ActiveX control pad property sheet, it doesn't create a new **Font** object and set it (like the property browser does). Instead, it inserts new values into the existing **Font** object. This means that the property **Set/Let** isn't called when the property is changed; instead only **Get** is called.

### **Workaround:**

To work around this problem, change the code in `UserControl_InitProperties` from:

```
Set Font = Ambient.Font
```

to:

```
Set Font.Name = Ambient.Font.Name  
Set Font.Size = Ambient.Font.Size  
(etc.)
```



## PropertyPage Object Does Not Support Activate and Deactivate Events (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscPropertyPagesActivateDeActivateEventsCantOccurReadMeC"}

The Activate and Deactivate events erroneously include the **PropertyPage** object in their Applies To lists.

## Visual Basic 5.0 Bidirectional Features (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscVisualBasic50BidirectionalFeaturesC;vbprolefttright"}

Visual Basic is bidirectional (also known as "BiDi")-enabled, but there is no mention of this on the product packaging or in any other printed documentation. Online documentation is contained in a separate help file included only with 32-bit bidirectional platforms.

"Bidirectional" is a generic term used to describe software products that support Arabic and other languages which are written right-to-left. More specifically, bidirectional refers to the product ability to manipulate and display text for both left-to-right and right-to-left languages. For example, displaying a sentence containing words written in both English and Arabic requires bidirectional capability.

Microsoft Visual Basic 5.0 includes standard features to create and run Windows applications with full bidirectional language functionality. However, these features are operational only when Microsoft Visual Basic 5.0 is installed in a bidirectional 32-bit Microsoft Windows environment, such as Arabic Microsoft Windows 95. Other bidirectional 32-bit Microsoft Windows environments are available as well.

The **RightToLeft** property has been added to forms, controls, and other Visual Basic objects to provide an easy mechanism for creating objects with bidirectional characteristics. Although **RightToLeft** is a part of every Microsoft Visual Basic 5.0 installation, it is operational only when Microsoft Visual Basic 5.0 is installed in a bidirectional 32-bit Microsoft Windows environment.

Bidirectional Features online help describes all Microsoft Visual Basic 5.0 bidirectional features. When Microsoft Visual Basic 5.0 is installed in a bidirectional 32-bit Microsoft Windows environment, choose the Bidirectional Features item on the Help menu to access Bidirectional Features online help. Otherwise, locate and double-click the file Bhelp32.hlp.

For compatibility with Microsoft Visual Basic 4.0, two versions of the 32-bit **Grid** control (Grid32.ocx) are included with Microsoft Visual Basic 5.0 but not installed. Both are located in the \Tools folder of the product media. The standard and bidirectional versions are located in the \Controls and \Controls\Bidi subfolders, respectively.

## VB5 Doesn't Update RDO Reference After Converting RDC (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See

Also": "rmmscVB5DoesntUpdateRDOReferenceAfterConvertingMSRDCControlReadMeC"}

If you convert a Visual Basic 4.0 project using an earlier version of Remote Data Objects (RDO), Visual Basic does not automatically switch the reference from the old version of RDO to the new version of RDO. To fix this, click References in the Project menu, deselect the old version of RDO and then select the new version of RDO.

## Bound Image Control Doesn't Display Picture if RDC Uses Batch Cursors (Readme)

{ewc HLP95EN.DLL,DYNALINK,"See

Also":"rmmscBoundImageControlDoesntDisplayPictureIfRDCUsesBatchCursorsReadmeC"}

If you have an **Image** control bound to an image-containing field in a **RemoteData** Control (RDC), and the RDC uses batch cursors (CursorDriver = rdUseClientBatch), the **Image** control doesn't display the image.

To correct this problem, set the RDC's **Options** property to 128 (rdFetchLongColumns).

## Upgrading VB Enterprise Edition Deletes Autmgr32.exe (ReadMe)

Removing the Enterprise edition of Visual Basic 4.0 after upgrading to the Enterprise edition of Visual Basic 5.0 deletes the Automation Manager. To avoid or fix this problem, either remove Visual Basic 4.0 prior to upgrading to VB 5.0, or reinstall Automation Manager from the Visual Basic 5.0 setup after Visual Basic 4.0 has been removed.

By default Visual Basic 5.0 is installed into the following directory: \Program Files\DevStudio\VB. Since this is a change from Visual Basic 4.0, which installed in its own directory (C:\VB in Windows NT 3.51 or C:\Program Files\Microsoft Visual Basic in Windows 95 and Windows NT 4.0), installing Visual Basic 5.0 does not automatically upgrade Visual Basic 4.0. You can run the two programs simultaneously.

In the Visual Basic 4.0 setup Automation Manager is not installed as a shared component. Therefore, it is removed when you remove Visual Basic 4.0. To reinstall the Automation Manager in Visual Basic 5.0, select Add/Remove Programs from the Control Panel, select Visual Basic 5.0 Enterprise Edition from the list of programs installed on your system, click the Add/Remove button, and then the Reinstall button.

## Unload Me Statement in ListBox Control ItemCheck Event Causes GPF (ReadMe)

Attempting to unload a form using `Unload Me` in the **ListBox** control's `ItemCheck` event causes a GPF. For example:

```
Private Sub Form_Load()  
    Dim i%  
    For i = 0 To 20  
        List1.AddItem i  
    Next i  
End Sub  
  
Private Sub List1_ItemCheck(Item As Integer)  
    Unload Me  
End Sub
```

This sample also produces a GPF if the spacebar is pressed, or if you attempt to select an item in the list box using a command button.

The recommended procedure for unloading a form is to add the **Unload** statement to the `Click` event in a **CommandButton** or **Menu** control.

## CommonDialog Control Constant cdlHelpContents May Be Obsolete (ReadMe)

Help files created in newer versions of the Windows Help Compiler may not support accessing Contents topics with the **CommonDialog** control constant `cdlHelpContents`. For example, attempting to access a help file Contents topic as in the example below may not produce the desired results:

```
Private Sub cmdHelpContents_Click()  
    dlgCommon.HelpFile = "help\vb5.hlp"  
    dlgCommon.HelpCommand = cdlHelpContents  
    dlgCommon.ShowHelp  
End Sub
```

Help files created with newer versions of the Windows Help Compiler use .cnt files to display the Contents topic in the Help viewer application. This is a change from previous versions. Therefore, the **CommonDialog** control constant `cdlHelpContents` no longer works.

The following information from the Platform SDK (formerly Win32 SDK) summarizes this change:

“In the past, applications have used `HELP_CONTENTS` and `HELP_INDEX` commands with the `WinHelp` function to display the Contents topic and the keyword index of the help file. These commands are no longer recommended. Use the `HELP_FINDER` command instead.”

Visual Basic 5.0 does not provide a **CommonDialog** control constant equivalent to the `WinHelp` function `HELP_FINDER` command. To access the Contents topic in a help file, declare the `WinHelp` command as a constant, as in this example:

```
Const HelpFinder = &h000B
```

For more information on using `WinHelp` commands, refer to the Platforms SDK.

## "Permission denied" or "Unexpected error" when compiling OCX or DLL (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscPermissionDeniedWithBinaryCompatibilityReadMeC"}

When compiling an ActiveX control or ActiveX DLL project with the Binary Compatibility option selected, you may encounter these errors if you:

- Compile the project to the same .ocx (or .dll) file you specified for Binary Compatibility, AND
- Compile the .ocx (or .dll) file more than once in the same Visual Basic session.

**Note** If the error log contains "LINK : fatal error LNK1104: cannot open file "<path>\<filename>" after you receive "Unexpected error occurred in code generator or linker," you must close Visual Basic and delete the file named in the message.

### Explanation

In general, you should not compile a component to the same file you specified for Binary Compatibility. Doing so will bulk up the file with extraneous interface information for these interim builds, as discussed in "Version Compatibility" in Books Online.

Instead, use the following rule of thumb to determine which Version Compatibility option to use:

- If you're creating a brand new control (or code component), select Project Compatibility and specify the file you normally build to.

**Note** Project Compatibility is the default for new projects.

- If you're creating a new version of a control (or code component) you have released to other developers, and you want your new version to be compatible with applications compiled using the earlier version, select Binary Compatibility and specify a reference copy of your released .ocx (or .dll) file.

**Important** This should be a copy of the file you distributed to developers; DO NOT specify the file you normally build the project to.

- If you're creating a new version of a control you have released to other developers, and you need to make changes that will prevent the control from working with previously compiled applications, select Project Compatibility and specify your normal build file.

Project Compatibility guarantees that copies of your control in existing projects can easily be upgraded when you distribute the new version to developers (hence the name). It does not guarantee that a new version of your control will be compatible with applications compiled using earlier versions.

**Important** Be sure to change the filename of your .ocx when you author a version that does not work with previously compiled applications. Standard practice is to include a version number in the filename. If you're creating a new version of a code component, you should also change the Project Name.

Version Compatibility options can be selected from the Component tab of the Project Properties dialog box.

**Note** "Permission denied" can also occur if the compiled .ocx file is in use by a client program. You must close all client programs — for example, Microsoft Internet Explorer or the Microsoft ActiveX Control Pad — in order to release the .ocx file so that you can recompile it.



## UserControl object doesn't support DDE (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscUserControlObjectDoesntSupportDDEDirectlyReadMeC"}

Controls authored with Visual Basic do not support Dynamic Data Exchange (DDE) to or from the **UserControl** object or any of its constituent controls.

**Note** The constituent controls on your **UserControl** will display DDE-related properties, methods, and events. However, if you attempt to set the **LinkMode** property, run-time error 369, "Operation not valid in an ActiveX DLL," will occur.

### More Information

The properties, methods, and events that allow a control to support DDE (such as the **LinkMode** property, or the LinkOpen event) are not part of the control. They are added by the container's Extender object, but only if the control's type library includes the necessary interfaces. These interfaces are not included in the type library created when an ActiveX control project is compiled, so the controls in the resulting .ocx file cannot support DDE.

Giving your control properties, methods, and events with the standard names will not enable DDE support for your control, because these properties and methods must be supplied by the container. In fact, if you supply events with the standard names (such as LinkOpen), Visual Basic will not allow you to put an instance of your control on a form.

## Changing UserControl property doesn't affect compiled applications (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See  
Also":"rmmscChangingControlToInvisibleAtRuntimeWontWorkInExistingHostsReadMeC"}

If you release a control authored with Visual Basic, and later decide to release an enhanced version of this control, some compatibility issues will not be handled by the Version Compatibility feature.

For example, consider the following scenario:

- You release SuperDuperComm.ocx version 1.0, and distribute it to developers. The control is visible at run time.
- Developers distribute applications compiled with SuperDuperComm.
- When you release version 2.0 of SuperDuperComm, you set the **InvisibleAtRuntime** property of the **UserControl** object to **True**, in response to developer feedback. You use Binary Compatibility (set using the Component tab of the Project Properties dialog box) to ensure that the new version will work with applications compiled with version 1.0.
- The new SuperDuperComm is installed on machines that have applications compiled with version 1.0, but developers report that the control is still visible at run time.

The reason for this is that invisibility at run time is a service provided by the container (for example, a Visual Basic form), and since version 1.0 of the control didn't ask for it, the compiled container doesn't supply it.

Other properties of the **UserControl** object that depend on services supplied by the container include:

- ControlContainer
- CanGetFocus
- Alignable
- ForwardFocus
- DefaultCancel
- ToolboxBitmap
- Public

Changing these properties from one version of your control to the next will not affect the control's behavior in compiled applications.

Related issues are discussed in "Versioning Issues for Controls," in Books Online.

## Event arguments that use OLE data types are not recognized in Access (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscColorPropertiesOfTypeOLECOLORAppearAsLongInAccessReadMeC"}

Microsoft Access does not recognize the standard OLE data types listed in the following table, and will translate them as shown:

<b>OLE data type</b>	<b>Appears in Access as</b>
OLE_TRISTATE	Long
OLE_HANDLE	Long
OLE_OPTEXCLUSIVE	Boolean
OLE_COLOR	0

For example, suppose you declare the following event in a control authored using Visual Basic:

```
Event TestType(ByVal OT As OLE_TRISTATE, _  
    ByVal OO As OPT_EXCLUSIVE, ByVal OC As OLE_COLOR)
```

If the compiled control is used in Microsoft Access, the event procedure will appear as follows:

```
Private Sub MyControl1_TestType(ByVal OT As Long, _  
    ByVal OO As Boolean, ByVal OC As 0)  
  
End Sub
```

For all types except OLE\_COLOR, the data types used by Microsoft Access are the same size as the standard data types. However, the code for OLE\_COLOR will not compile. Thus, if you expect your control to be used in Microsoft Access, do not use OLE\_COLOR as a data type for event arguments.

## UserControl containing User Forms Textbox may not work on a UserForm (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See  
Also":"rmmscUserControlContainingUserFormsTextboxMayNotWorkOnUserFormReadMeC"}

The Microsoft User Forms used in Microsoft Office applications include a set of controls similar to the intrinsic controls in Visual Basic. These controls can be used on Visual Basic forms, or placed on **UserControl** or **UserDocument** designers.

**Note** If you use these controls in your application or .ocx, SetupWizard must include the support DLL for Microsoft User Forms in your Setup program.

The following scenario is known to cause problems:

- Author a control in Visual Basic by placing a User Forms Textbox on a **UserControl** designer, and then make an .ocx file.
- Place an instance of your new control on a User Form.
- When the form is run, it may not be possible to type in the User Forms Textbox.

Use of the User Forms Textbox control on a **UserControl** object is not recommended.

## An object may terminate with a Friend method in the Call Tree (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscAnObjectMayTerminateWithFriendMethodInCallTreeReadMeC"}

When you call a property or method of an object, that procedure remains in the Call Tree (that is, on the stack) until it returns. If all references to an object are released while one of its public properties or methods is in the Call Tree, Visual Basic will not allow the object to terminate. The object's Terminate event will only occur after it has no more public properties or methods in the Call Tree.

Visual Basic provides this protection for public members because if an object is allowed to terminate while code is in the Call Tree, the memory used by the object will be freed, and a program fault will occur when the member finally returns.

Properties and methods declared with the **Friend** keyword do not have this protection. In most cases it is not necessary, because in ordinary programming scenarios with **Friend** members it is very difficult to cause an object to terminate while it has code on the stack.

One scenario in which this may occur is when a **Friend** method is used to raise a public event, AND the **Friend** method is called by an event outside Visual Basic, such as a system timer set up using the **AddressOf** operator. In this case, the method must be changed from **Friend** to **Public** to avoid potential program faults.

The XTimer class used in the Coffee sample application provides an example of this.

## Manually editing VB\_PredeclaredId attribute causes problems (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscManuallyEditingVB\_PredeclaredIdAttributeCausesProblemsReadMeC"}

If you edit Visual Basic source files using a text or code editor other than Visual Basic, be careful not to change settings of the attribute VB\_PredeclaredId. In particular, changing this attribute may cause serious problems with GlobalMultiUse and GlobalSingleUse classes.

In general, you should not edit attributes manually, as doing so may put the module into an internally inconsistent state.

## AmbientProperties.Font clones container's ambient font (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscAmbientPropertiesFontClonesContainersAmbientFontReadMeC"}

When you access the **Font** property of the **AmbientProperties** object, you do not get a reference to the container's font. Rather, you get a copy (clone) of the font.

The reason for this is that **AmbientProperties.Font** is commonly used to initialize a control's font whenever an instance of the control is added to a container. If a reference to the font itself were supplied, then changing the control's font would change the container's font as well.

If for some reason your control requires a reference to the container's font, you can obtain it by accessing the container through the **Parent** property of the **UserControl** object.

## To use GlobalMultiUse objects internally, declare a global variable (ReadMe)

```
{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscToUseGlobalMultiUseObjectsInternallyDeclareGlobalVariableReadMeC"}
```

Setting the **Instancing** property of a class to GlobalMultiUse or GlobalSingleUse in an ActiveX Exe or ActiveX DLL project adds the properties and methods of that class to the global name space of any client project. That is, the properties and methods can be used as if they were global functions, without explicitly creating an instance of the class.

The creation and use of such global objects is discussed in "Global Objects and Code Libraries," in Books Online. This topic also discusses the limitations of global objects, in particular the following limitation:

"The properties and methods of a global object only become part of the global name space when the component is referenced from *other projects*. Within the project where you created the GlobalMultiUse class module, objects created from the class are just ordinary objects."

That is, if you create project MyUtilities that contains a Utilities class with **Instancing = GlobalMultiUse**, and give the class an InvertMatrix method, then from any client project (one in which you've set a reference to MyUtilities) you can call InvertMatrix without any qualification. However, within the MyUtilities project you must create an object of type Utilities and use that object to call InvertMatrix.

The recommended workaround (which is only hinted at in the topic noted above) is to declare a global variable in a standard module, as follows:

```
Public Utilities As New Utilities
```

Thereafter, whenever you need to use InvertMatrix (or any other procedure supplied by the Utilities class), you can qualify it with the class name:

```
Utilities.InvertMatrix aintMyLargeMatrix
```

The first time you use a method of the Utilities class in this fashion, an instance of the class is created automatically, because the global variable is declared **As New**. Using the class name as the name of the variable makes it clear which of the modules within your component is supplying the procedure.

**Note** You must declare this global variable in a standard module, not a class module, in order for this technique to work.



## Component that uses App.PrevInstance can cause problems (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscAPEComponentThatUsesAppPrevInstanceCanCauseProblemsReadMeC"}

A computer running Windows NT can support multiple desktops. When you're using a component designed to work with distributed COM, this can result in the following scenario:

- A client program in a user desktop requests one of the objects the component provides. Because the component is physically located on the same machine, the component is started in the user desktop.
- Subsequently, a client program on another computer uses distributed COM to request one of the objects the component provides. A second instance of the component is started, in a system desktop.

There are now two instances of the component running on the same NT computer, in different desktops.

This scenario is not a problem, unless the author of the component has placed a test for **App.PrevInstance** in the startup code for the component, to prevent multiple copies of the component from running on the same computer. In this case, the remote object creation will fail.

## Don't change font on multiple controls with floating Properties window (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See

Also": "rmmscDontChangeFontOnMultipleControlsWithFloatingPropertiesWindowReadMeC"}

When using Visual Basic in Multiple Document Interface (MDI) mode, it is possible to display the Properties window either docked to the main window (the default) or floating.

If the Properties window is floating, AND two or more controls are selected, AND you change the Font property of the controls using the floating Properties window, a program fault will occur in Visual Basic.

## Don't change default member to Friend or Private (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscDontChangeDefaultMemberToFriendOrPrivateReadMeC"}

If you have used the Procedure Attributes dialog box to make a public property or method of a class the default member for that class, do not edit the declaration for the member and change **Public** to **Friend** or **Private**, without first removing the Default attribute. Doing so will cause serious problems in your program.

If you find yourself in this situation, make the member **Public** again, and use the Procedure Attributes dialog to remove the Default attribute. You can then change the declaration back to **Friend** or **Private**.

## Help for DateSerial gives incorrect information on Year argument (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscHelpForDateSerialGivesIncorrectInformationOnYearArgumentReadMeC"}

Help for the **DateSerial** function incorrectly describes the way the **year** argument is interpreted. The text should read as follows:

For the **year** argument, values between 0 and 29, inclusive, are interpreted as the years 2000–2029. Values between 30 and 99, inclusive, are interpreted as the years 1930–1999. For all other year arguments, use a four-digit year (for example, 1800).

## If you're using Visual Basic 5.0 with Windows NT 4.0... (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmsclfYoureUsingVisualBasic50WithWindowsNT40ReadMeC"}

Microsoft regularly provides updates to Windows NT, to supply customers with the latest corrections to problems that have been reported. These *Service Packs* are referred to by the letters "SP" followed by a number (for example, "SP1").

When you install Visual Basic 5.0 on a system running Windows NT 4.0, Visual Basic automatically installs the OLE Automation support DLLs included in SP2. The entire SP2 update is included on the distribution CD, and it is highly recommended that you apply it to your system.

**Note** If the application you're creating is to run on Windows NT 4.0, users should upgrade their systems to SP2 before installing your application. This is particularly important if your application uses distributed COM.

The following problems are known to occur with SP2, and are fixed in SP3. If you have applied SP2 to your system, and you encounter any of these problems, you should obtain and apply SP3.

- OLE initialization can fail if an object is created on a second thread while the first thread is still initializing OLE. This is most likely to occur when running multithreaded Exe components in stressful conditions (such as multiprocessor machines).
- Calling a method of an object in a component running on another machine, using distributed COM, fails if the method has more than 16 arguments, and at least one but not more than 15 of those arguments are ByRef.
- OLEAUT32.DLL fails to self-register if StdFont and StdPicture are not correctly registered (RISC only).
- Alignment fault when loading a metafile into a Picture object (RISC only).
- Problems displaying GIF and JPEG files in 16-color mode.
- Font facenames longer than 16 characters can overwrite data on the stack (RISC only).
- When a double-byte Font is copied after one or more properties have been changed, the cloned Font may be in an inconsistent state. (A workaround is to read one property before copying the Font.)
- Palette behavior is inconsistent with Visual Basic 4.0.
- A program fault can occur when using controls that display property pages (RISC only).
- Problems displaying double-byte fonts (for example, Japanese).
- When using distributed COM between a Windows NT system and a UNIX system, Variants will not be passed correctly.

**Note** SP3 is not available on the VB5 CD. Information on its availability can be obtained from the Microsoft Visual Basic Web Site. As with SP2, users should upgrade their systems to SP3 before installing your application.

Some of the items in the list above also affect Visual Basic applications running on Windows 95. SP3 cannot be used to upgrade Windows 95. Microsoft is currently investigating distribution mechanisms for operating systems other than Windows NT; information will be available on the Microsoft Visual Basic Web Site.

## MaskPicture property requires black and white bitmap in Windows 95 (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscMaskColorPropertyRequiresBlackWhiteBitmapInWindows95ReadMeC"}

When a multicolor bitmap is assigned to the **MaskPicture** property of a **UserControl**, and a color other than white is assigned to the **MaskColor** property, the control will behave as if the **MaskColor** property had been set to white.

This problem occurs when the control is used on systems running Windows 95. To correct the problem, use a bitmap with two colors — white for all areas where the control is to be transparent, and black for all areas where the control is to be opaque.

**Note** Set the **MaskColor** to white, so that transparency will also work correctly on systems running Windows NT.

A color bitmap can be converted to a black-and-white bitmap by changing the pixels as follows: If a pixel is the desired mask color, set it to white; if the pixel is any other color, set it to black.

## Exe component hangs creating object from second Exe during startup (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscActiveXExeComponentHangsIfInstalledOnWindows95ReadMeC"}

If the startup code for an out-of-process component (ActiveX Exe) attempts to create an object from a second out-of-process component, before returning control to the client Exe, the component will hang.

This occurs only on systems running Windows 95.

The component will hang only if all three programs — the client and both components — are single-threaded .exe files. You can avoid the problem by deferring creation of the object supplied by the second component until a later time.

The problem can also be fixed by upgrading the Windows 95 machine with the Distributed COM support package for Windows 95.

## StandardSize Property for Property Pages Doesn't Work As Expected (Readme)

{ewc HLP95EN.DLL,DYNALINK,"See  
Also":"rmmscStandardSizePropertyForPropertyPagesDoesntWorkAsExpectedReadmeC"}

The **StandardSize** property for property pages may not give the expected result when set to 1 - Small or 2 - Large. This property only affects the size of the property page for the author's display driver and resolution. Unexpected results may occur on other displays.

It is suggested that you use the default **StandardSize** property value (1 - Custom) and treat a property page as you would any other form.



## PaintPicture Method: Opcode Argument Only Applies to Bitmaps (Readme)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscPaintPictureMethodOpcodeArgumentOnlyAppliesToBitmapsReadmeC"}

*Opcode*, the last optional argument to the PaintPicture method, is used to pass a bitwise operation on a bitmap. Placing a value in this argument when passing other image types will cause a "Invalid procedure call or argument" error. This is by design. To avoid this error, leave the *Opcode* argument blank for any image other than a bitmap.

## Help Button Doesn't Show Up On Property Page (Readme)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscHelpButtonDoesntShowUpOnPropertyPageReadmeC"}

When working with a UserControl in the design environment, the Help button on the property page may not show up the first time the property page is shown. To remedy the situation, open and then close the designer for the UserControl. The next time the property page is accessed the Help button will appear.

This is a known problem only in the design environment; Property pages for a compiled UserControl do not exhibit this behavior.

## Referencing a Private UserControl Before Form Load May Cause Error (Readme)

{ewc HLP95EN.DLL,DYNALINK,"See  
Also":"rmmscReferencingPrivateUserControlPriorToFormLoadCausesTypeMismatchErrorReadmeC"}

A "Type Mismatch" error may occur if you try to reference a **UserControl** before loading the containing form. For example, the **Set** statement in the following code causes an error:

```
Sub Main()  
    Dim a As UserControl1  
    Set a = Form1.UserControl11  
End Sub
```

To prevent this error, load the form before referencing the **UserControl**:

```
Sub Main()  
    Dim a As UserControl1  
    Form1.Show  
    Set a = Form1.UserControl11  
End Sub
```

This behavior only occurs with **UserControls** added to a project via the Add UserControl command on the Project menu, and with the **Public** property set to **False**. Public **UserControls** added via the References dialog do not cause the error.

## Changing Case of Form or Module Names May Cause Load Error (Readme)

Changing the case of the **Name** property value for a Form or other module without otherwise changing the name itself can cause a “Conflicting names” error message the next time the project containing the is loaded. For example, changing “Form1” to “form1” will cause the error; changing “Form1” to “formX” will not.

The error is caused by the way module names are stored within the project file – the procedure for changing names within the project file isn’t case sensitive, while the procedure for reading names on project load is.

## Control Creation Edition Installs Some Files as Read-only (Readme)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rmmscControlCreationEditionInstallSomeFilesAsReadonlyReadmeC"}

Some of the documentation files included with the Control Creation Edition of Visual Basic are installed with read-only attributes. This can cause a "Read-only" warning to be displayed when attempting to uninstall Visual Basic.

To avoid the warnings, change the attributes of any read-only files in the Visual Basic directory before uninstalling.

## Hard-typed Objects Permitted as Optional Parameters (ReadMe)

The description of permitted types in optional arguments is incorrect. An argument specified as **Optional** can have a specific object type. For example, the argument specification **Optional MyArg As Worksheet** is valid as long as the object type is within scope of the declaration. The only type prohibited is a user-defined type.

## Method Incorrectly Called Property (ReadMe)

**DesignerWindow** is referred to as a property in its help topic title. It is actually a method. The substantive information in the topic is correct.

## Properties Incorrectly Called Methods (ReadMe)

The following Add-in elements are referred to as methods in their help topic titles. They are actually properties: **ProcBodyLine**, **ProcCountLines**, **ProcOfLine**, **ProcStartLine**. The substantive information in these topics is correct.



## Missing Property Topics (RDO) (ReadMe)

{ewc HLP95EN.DLL,DYNALINK,"See Also":""}

The following RDO properties do not have Help topics:

**rdoQueries** Property

**rdoResultSets** Property

**rdoTables** Property

**rdoConnections** Property

**rdoColumns** Property

**rdoParameters** Property

**rdoEnvironments** Property

These properties are used to access collections of the corresponding RDO objects. For example, the **rdoQuery** Object has an **rdoParameters** property. This property is simply the collection of **rdoParameter** objects that apply to that query. Both the collections and the underlying objects for these properties are fully documented in online Help.

## Error in "Inserting a User Connection Object"

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjConnection"}

The fourth step in the procedure at the beginning of "Inserting a New User Connection Object," in the Guide to Building Client/Server Applications With Visual Basic is incorrect and should be ignored. The correct procedure is completely described by the first three steps.

## Status Property (RDO) Applies to Both Rows and Columns (Readme)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstStatusPropertyRdoColumnObject"}

The text of the Help topic for the RDO Status property only mentions Rows, but the property applies to both rows and columns.

