Data Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjDatabase;daobjDataC;daobjRecordset;daproConnect;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"daobjDataX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"daobjDataP;daproConnect;daproEditMode"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"daobjDataM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"daobjDataE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"daobjDataS"}

Provides access to data stored in databases using any one of three types of **Recordset** objects. The **Data** control enables you to move from record to record and to display and manipulate data from the records in bound controls. Without a **Data** control or an equivalent data source control like the **RemoteData** control, data-aware (bound) controls on a form can't automatically access data.

Syntax

Data

Remarks

You can perform most data access operations using the **Data** control without writing any code at all. Data-aware controls bound to a **Data** control automatically display data from one or more fields for the current record or, in some cases, for a set of records on either side of the current record. The **Data** control performs all operations on the current record.

If the **Data** control is instructed to move to a different record, all bound controls automatically pass any changes to the **Data** control to be saved in the database. The **Data** control then moves to the requested record and passes back data from the current record to the bound controls where it's displayed.

The **Data** control automatically handles a number of contingencies including empty recordsets, adding new records, editing and updating existing records, and handling some types of errors. However, in more sophisticated applications, you need to trap some error conditions that the **Data** control can't handle. For example, if the Microsoft Jet database engine has a problem accessing the database file, doesn't have permission, or can't execute the query as coded, a trappable error results. If the error occurs before your application procedures start or due to some internal errors, the Error event is triggered.

Bound Controls

The **DBList**, **DBCombo**, **DBGrid**, and **MSFlexGrid** controls are all capable of managing sets of records when bound to a **Data** control. All of these controls permit several records to be displayed or manipulated at once.

The intrinsic **Picture**, **Label**, **TextBox**, **CheckBox**, **Image**, **OLE**, **ListBox** and **ComboBox** controls are also data-aware and can be bound to a single field of a **Recordset** managed by the **Data** control. Additional data-aware controls like the **MaskedEdit** and **RichTextBox** controls are available in the Professional and Enterprise Editions and from third-party vendors.

Operation

Once the application begins, Visual Basic uses **Data** control properties to open the selected database, create a **Database** object and create a **Recordset** object. The **Data** control's **Database** and **Recordset** properties refer to the newly created **Database** and **Recordset** objects which may be manipulated independently of the **Data** control — with or without bound controls. The **Data** control is initialized *before* the initial Form_Load event for the form on which it is placed. If any errors occur during this initialization step a non-trappable error results.

When Visual Basic uses the Jet database engine to create a **Recordset**, no other Visual Basic operations or events can occur until the operation is complete. However, other Windows-based applications are permitted to continue executing while the **Recordset** is being created. If the user

presses CTRL+BREAK while the Jet engine is building a **Recordset**, the operation is terminated, a trappable error results, and the **Recordset** property of the **Data** control is set to **Nothing**. In design time, a second CTRL+BREAK causes Visual Basic to display the Debug window.

When you use a **Data** control to create a **Recordset** object or when you create a **Recordset** object in code and assign it to the **Data** control, the Microsoft Jet database engine automatically populates the **Recordset** object. As a result, bookmarks (and for snapshot-type **Recordset** objects, recordset data) are saved in local memory; the user doesn't need to manipulate the **Data** control, and you don't need to invoke the **MoveLast** method in code. Page locks used to create the **Recordset** are released more quickly, making it possible for other **Recordset** objects to access the same data. **Recordset** objects created in code but not assigned to the **Data** control aren't automatically populated by the Jet engine. Populate these objects through code. Because of the way that the **Data** control populates its **Recordset** in the background, an additional cloned **Recordset** might be created.

You can manipulate the **Data** control with the mouse, moving from record to record or to the beginning or end of the **Recordset**. The **EOFAction** and **BOFAction** properties determine what happens when the user moves to the beginning or end of a **Recordset** with the mouse. You can't set focus to the **Data** control.

Validation

Use the Validate event and the **DataChanged** property to perform last minute checks on the records being written to the database.

Data Access Objects

You can use the **Database** and **Recordset** data access objects created by the **Data** control in your procedures. The **Database** and **Recordset** objects each have properties and methods of their own, and you can write procedures that use these properties and methods to manipulate your data.

For example, the **MoveNext** method of a **Recordset** object moves the current record to the next record in the **Recordset**. To invoke this method, you could use this code:

```
Data1.Recordset.MoveNext
```

The **Data** control is capable of accessing any of the three types of Jet engine Version 3.0 **Recordset** objects. If you don't select a recordset type, a dynaset-type **Recordset** is created.

In many cases, the default type and configuration of the **Recordset** object created is extremely inefficient. That is, you might not need an updatable, fully-scrollable, keyset-type cursor to access your data. For example, a read-only, forward-only, snapshot-type **Recordset** might be far faster to create than the default cursor. Be sure to choose the most efficient **Type**, **Exclusive**, **Options** and **ReadOnly** properties possible for your situation.

Note The constants used to request a specific **Recordset** type when using the **Data** control are different than the constants used to determine the type of **Recordset** created or to create a **Recordset** using the **OpenRecordset** method.

To select a specific type of **Recordset**, set the **Data** control's **RecordsetType** property to:

Recordset Type	Value	Constant
Table	0	**vbRSTypeTable**
Dynaset	1	(Default) **vbRSTypeDynaset**
Snapshot	2	**vbRSTypeSnapshot**

Important The **Data** control cannot be used to access **Recordset** objects created with the **dbForwardOnly** option bit set.

Professional and Enterprise Editions

As far as data access is concerned, the primary difference between the Standard, Professional and

Enterprise Editions of Visual Basic is the ability to create new data access objects. In the Standard Edition, you can't declare (with the **Dim** keyword) variables as data access objects in code. This means that only the **Data** control can create **Database** and **Recordset** objects.

In Visual Basic version 5.0 Professional and Enterprise Editions, you can create a new **Recordset** object and assign it to the **Data** control's **Recordset** property. Any bound controls connected to the **Data** control permit manipulation of the records in the **Recordset** you created. Make sure that your bound controls' **DataField** properties are set to field names that are valid in the new **Recordset**.

Stored Queries

Another important option when using the **Data** control is the ability to execute stored queries. If you create a **QueryDef** object beforehand, the **Data** control can execute it and create a **Recordset** using the **QueryDef** object's stored **SQL**, **Connect** and other properties. To execute a **QueryDef**, set the **Data** control's **RecordSource** property to the **QueryDef** name and use the **Refresh** method.

If the stored **QueryDef** contains parameters, you need to create the **Recordset** and pass it to the **Data** control.

BOF/EOF Handling

The **Data** control can also manage what happens when you encounter a **Recordset** with no records. By changing the **EOFAction** property, you can program the **Data** control to enter AddNew mode automatically.

You can program the **Data** control to automatically snap to the top or bottom of its parent form by using the **Align** property. In either case, the **Data** control is resized horizontally to fill the width of its parent form whenever the parent form is resized. This property allows a **Data** control to be placed on an MDI form without requiring an enclosing **Picture** control.

Data Control Constants

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

Error Event Constants

Constant	Value	Description
vbDataErrContinue	0	Continue
vbDataErrDisplay	1	(Default) Display the error message

Validate Event Action Constants

Constant	Value	Description
vbDataActionCancel	0	Cancel the operation when the **Sub** exits
vbDataActionMoveFirst	1	**MoveFirst** method
vbDataActionMovePrevious	2	**MovePrevious** method
vbDataActionMoveNext	3	**MoveNext** method
vbDataActionMoveLast	4	**MoveLast** method
vbDataActionAddNew	5	**AddNew** method
vbDataActionUpdate	6	Update operation (not **UpdateRecord**)
vbDataActionDelete	7	**Delete** method
vbDataActionFind	8	**Find** method
vbDataActionBookmark	9	The **Bookmark** property is set
vbDataActionClose	10	**Close** method
vbDataActionUnload	11	The form is being unloaded

Beginning-Of-File Action Constants

Constant	Value	Description
vbMoveFirst	0	Move to first record
vbBOF	1	Move to beginning of file

End-Of-File Action Constants

Constant	Value	Description
vbMoveLast	0	Move to last record
vbEOF	1	Move to end of file
vbAddNew	2	Add new record to end of file

Recordset-Type Constants

Constant	Value	Description
vbRSTypeTable	0	Table-type recordset
vbRSTypeDynaset	1	Dynaset-type recordset
vbRSTypeSnapShot	2	Snapshot-type recordset

Error Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daevtErrorC;damthAddNew;damthDelete;daobjRecordset;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"daevtErrorX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"daevtErrorA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"daevtErrorS"}

Occurs only as the result of a data access error that takes place when no Visual Basic code is being executed.

Syntax

Private Sub *object*_**Error (**[*index* **As Integer**,] *dataerr* **As Integer**, *response* **As Integer)**

The Error event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	Identifies the control if it's in a control array.
dataerr	The error number.
response	A number corresponding to the response you want to take, as described in Settings.

Settings

The settings for *response* are:

Constant	Value	Description
vbDataErrContinue	0	Continue
vbDataErrDisplay	1	(Default) Display the error message

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

You usually provide error-handling functionality for run-time errors in your code. However, run-time errors can occur when none of your code is running, as when:

- A user clicks a **Data** control button.
- The **Data** control automatically opens a database and loads a **Recordset** object after the Form_Load event.
- A custom control performs an operation such as the **MoveNext** method, the **AddNew** method, or the **Delete** method.

If an error results from one of these actions, the Error event occurs.

If you don't code an event procedure for the Error event, Visual Basic displays the message associated with the error.

Errors that occur *before* the Form_Load event, are not trappable and do not trigger the Error event. For example, if at design time you set the properties of the Data control to point to an unknown database table an untrappable error results.

Error Event Example

This example displays an Open dialog box if the database specified in the **Data** control's **DatabaseName** property isn't found *after* the Form_Load event is complete.

```
Private Sub Data1_Error (DataError As Integer, Response As Integer)
   Select Case DataError
      ' If database file not found.
      Case 3024
         ' Display an Open dialog box.
         CommonDialog1.ShowOpen
      ...
   End Select
End Sub
```

Reposition Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daevtRepositionC;damthFindFirst;daobjRecordset;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"daevtRepositionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"daevtRepositionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"daevtRepositionS"}

Occurs after a record becomes the current record.

Syntax

Private Sub *object*.**Reposition (**[*index* **As Integer**]**)**

The Reposition event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	Identifies the control if it's in a control array.

Remarks

When a **Data** control is loaded, the first record in the **Recordset** object becomes the current record, causing the Reposition event. Whenever a user clicks any button on the **Data** control, moving from record to record, or if you use one of the Move methods, such as **MoveNext**, the Find methods, such as **FindFirst**, or any other property or method that changes the current record, the Reposition event occurs after each record becomes current.

In contrast, the **Validate** event occurs before moving to a different record.

You can use this event to perform calculations based on data in the current record or to change the form in response to data in the current record.

Reposition Event Example

This example uses the Reposition event to update a list of Titles from the selected Publisher in the Biblio.mdb sample database. First place a **DBGrid**, **TextBox**, and two **Data** controls on a form. Set the **DatabaseName** properties of both **Data** controls to the Biblio.mdb sample database. Set the **RecordSource** property of Data1 to **Publishers**. Set the **DataSource** property of Text1 to **Data1** and the **DataField** property to **Name**. Set the **DataSource** property of DBGrid1 to **Data2**. Add the following code:

```
Private Sub Data1_Reposition()
   ' Select all Titles that are published by
   ' the current record in Data1
   Data2.RecordSource = "Select * from Titles where PubID = " &
Data1.Recordset("PubID")
   Data2.Refresh ' Rebuild the recordset
End Sub
```

The list of Titles in the **DBGrid** control will automatically be updated as you move through the Publishers recordset with Data1.

Validate Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"damthAddNew;damthClose;damthDelete;damthEdit;damthUpdate;vbevtValidateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"daevtValidateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtValidateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtValidateS"}

Occurs before a different record becomes the current record; before the **Update** method (except when data is saved with the **UpdateRecord** method); and before a **Delete**, **Unload**, or **Close** operation.

Syntax

Private Sub *object*_**Validate (**[*index* **As Integer**,] *action* **As Integer**, *save* **As Integer)**

The Validate event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	Identifies the control if it's in a control array.
action	An integer that indicates the operation causing this event to occur, as described in Settings.
save	A Boolean expression specifying whether bound data has changed, as described in Settings.

Settings

The settings for *action* are:

Constant	Value	Description
vbDataActionCancel	0	Cancel the operation when the **Sub** exits
vbDataActionMoveFirst	1	**MoveFirst** method
vbDataActionMovePrevious	2	**MovePrevious** method
vbDataActionMoveNext	3	**MoveNext** method
vbDataActionMoveLast	4	**MoveLast** method
vbDataActionAddNew	5	**AddNew** method
vbDataActionUpdate	6	**Update** operation (not **UpdateRecord**)
vbDataActionDelete	7	**Delete** method
vbDataActionFind	8	**Find** method
vbDataActionBookmark	9	The **Bookmark** property has been set
vbDataActionClose	10	The **Close** method
vbDataActionUnload	11	The form is being unloaded

The settings for *save* are:

Setting	Description
True	Bound data has changed
False	Bound data has not changed

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

The *save* argument initially indicates whether bound data has changed. This argument can still be **False** if data in the copy buffer is changed. If *save* is **True** when this event exits, the **Edit** and **UpdateRecord** methods are invoked. Only data from bound controls or from the copy buffer where the **DataChanged** property is set to **True** are saved by the **UpdateRecord** method.

This event occurs even if no changes have been made to data in bound controls and even if no bound controls exist. You can use this event to change values and update data. You can also choose to save data or stop whatever action is causing the event to occur and substitute a different action.

You can change the *action* argument to convert one action into another. 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). When using **AddNew**, you can use **MoveNext** and then execute another **AddNew** to examine the **EditMode** property to determine if an **Edit** or **AddNew** operation is in progress. Attempting to change **AddNew** or one of the Moves into any of the other actions is either ignored or produces a trappable error. Any action can be stopped by setting *action* to 0.

In your code for this event, you can check the data in each bound control where **DataChanged** is **True**. You can then set **DataChanged** to **False** to avoid saving that data in the database.

You can't use any methods (such as **MoveNext**) on the underlying **Recordset** object during this event.

AfterColUpdate Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daevtAfterColUpdateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"daevtAfterColUpdateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"daevtAfterColUpdateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"daevtAfterColUpdateS"}

Occurs after data is moved from a cell in the **DBGrid** control to the control's copy buffer.

Syntax

Private Sub *object*_**AfterColUpdate (**[*index* **As Integer**,] *colindex* **As Integer)**

The AfterColUpdate event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that identifies a control if it is in a control array.
colindex	An integer that identifies the column in the control.

Remarks

When a user completes editing within a **DBGrid** control cell, as when tabbing to another column in the same row, pressing ENTER, or when the control loses focus, the BeforeColUpdate event is executed, and unless canceled, data from the cell is moved to the control's copy buffer. Once moved, the AfterColUpdate event is executed.

The AfterColUpdate event occurs after the BeforeColUpdate event, and only if the *cancel* argument in the BeforeColUpdate event is not set to **True**.

Once the AfterColUpdate event procedure begins, the cell data has already been moved to the control's copy buffer and can't be canceled, but other updates can occur before the data is committed to the **Recordset**.

AfterColUpdate Event Example

This example does a lookup when one column is updated and places the result in another column.

```
Private Sub DataGrid1_AfterColUpdate (ColIndex As Integer)
   If ColIndex = 1 Then
      Data1.Recordset.FindFirst "PubId = " _
       & DataGrid1.Columns(1).Value
      If Not Data1.Recordset.NoMatch Then
         DataGrid1.Columns(2).Value = _
          Data1.Recordset.Fields("Publisher")
      Else
         DataGrid1.Columns(2).Value = "No Match"
      End If
   End If
End Sub
```

AfterDelete Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daevtAfterDeleteC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"daevtAfterDeleteX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"daevtAfterDeleteA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"daevtAfterDeleteS"}

Occurs after the user deletes a selected record in the **DBGrid** control.

Syntax

Private Sub *object*_**AfterDelete (**[*index* **As Integer**,] *colindex* **As Integer)**

The AfterDelete event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that identifies a control if it is in a control array.
colindex	An integer that identifies the column.

Remarks

When the user selects a record selector in the **DBGrid** control and presses DEL or CTRL+X, the selected row is deleted. Before the record is deleted, the BeforeDelete event is triggered. Once the row is deleted, the AfterDelete event is triggered. The row selected for deletion is available in the collection provided by the **SelBookmarks** property.

AfterDelete Event Example

This example displays a message confirming that a record has successfully been deleted.

```
Private Sub DataGrid1_AfterDelete ()
   MsgBox "Record has successfully been deleted!"
End Sub
```

AfterInsert Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daevtAfterInsertC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"daevtAfterInsertX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"daevtAfterInsertA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"daevtAfterInsertS"}

Occurs after the user inserts a new record into the **DBGrid** control.

Syntax

Private Sub *object*_**AfterInsert (***index* **As Integer)**

The AfterInsert event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that identifies a control if it is in a control array.

Remarks

When the user selects the new record (at the bottom of the control) and enters a character in one of the cells, the BeforeInsert event is triggered, followed by the BeforeUpdate, AfterUpdate and AfterInsert events.

When the AfterInsert event is triggered, the record has already been added to the database. The **Bookmark** property can be used to access the new record.

The AfterInsert event can't be canceled.

The AfterInsert event procedure can be used to update other tables or to perform post-update cleanup of other controls.

AfterInsert Event Example

This example creates an entry in a related table if the user enters a value in a column in the grid.

```
Private SubDataGrid1_AfterInsert ()
   If DataGrid1.Columns(1).Value <> "" Then
      Data2.Recordset.AddNew
      Data2.Recordset.Fields("PubId") = DataGrid1.Columns(1).Value
      Data2.Recordset.Update
   End If
End Sub
```

AfterUpdate Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daevtAfterUpdateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"daevtAfterUpdateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"daevtAfterUpdateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"daevtAfterUpdateS"}

Occurs after changed data has been written to the database from a **DBGrid** control.

Syntax

Sub *object*_**AfterUpdate (***index* **As Integer)**

The AfterUpdate event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that identifies a control if it is in a control array.

Remarks

When the user moves to another row, or the **Recordset** object's **Update** method is executed, data is moved from the control's copy buffer to the **Data** control's copy buffer and written to the database. Once the write is complete, the AfterUpdate event is triggered.

The updated record is available by using the **Bookmark** property of the **DBGrid** control.

The AfterUpdate event occurs after the BeforeUpdate event, but before the LostFocus event for the control (or GotFocus for the next control in the tab order). This event occurs in bound and unbound mode and can't be canceled.

Unlike the Change event, changing data in a control or record using code doesn't trigger this event.

AfterUpdate Event Example

This example updates a label when any change has been made in the grid.

```
Private Sub DataGrid1_AfterUpdate ()
   Label1.Caption = "Last modified: " & Format$(Now, "Long Date")
End Sub
```

BeforeColUpdate Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daevtBeforeColUpdateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"daevtBeforeColUpdateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"daevtBeforeColUpdateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"daevtBeforeColUpdateS"}

Occurs after editing is completed in a cell, but before data is moved from the cell to the **DBGrid** control's copy buffer.

Syntax

Private Sub *object*_**BeforeColUpdate (**[*index* **As Integer**,] *colindex* **As Integer**, *oldvalue* **As Variant**, *cancel* **As Integer)**

The BeforeColUpdate event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that identifies a control if it is in a control array.
colindex	An integer that identifies the column.
oldvalue	A value that contains the value contained in the cell prior to the change.
cancel	A Boolean expression expression that specifies whether the change occurs, as described in Settings.

Settings

The settings for *cancel* are:

Setting	Description
True	Cancels the change, restores cell to *oldvalue*, and restores focus to the control.
False	(Default) Continues with change and permits change of focus.

Remarks

The data specified by the o*ldvalue* argument moves from the cell to the control's copy buffer when a user completes editing within a cell, as when tabbing to another column in the same row, pressing ENTER, or changing focus away from the cell. Before the data has been moved from the cell into the control's copy buffer, the BeforeColUpdate event is triggered. This event gives your application an opportunity to check the individual grid cells before they are committed to the control's copy buffer.

If your event procedures set the *cancel* argument to **True**, the previous value is restored in the cell and focus remains on the control and the AfterColUpdate event is not triggered.

To restore *oldvalue* in the cell and permit the user to move focus off of the cell, set *cancel* to **False** and set the cell to *oldvalue* as follows:

```
Cancel = False
DBGrid1.Columns(ColIndex).Value = OldValue
```

The AfterColUpdate event occurs after the BeforeColUpdate event.

By setting the *cancel* argument to **True**, the user can not move the focus from the control until the application determines that the data can be safely moved back to the control's copy buffer.

BeforeColUpdate Event Example

This example checks to make sure that the value the user has typed in is within a certain range; otherwise it disables the update.

```
Private Sub DataGrid1.BeforeColUpdate (ColIndex As Long, OldValue As
Variant, Cancel As Integer)
   If ColIndex = 1 Then
      If DataGrid1.Columns(1).Value < Now Then
         Cancel = True
         MsgBox "You must enter a date that is later than today."
      End If
   End If
End Sub
```

BeforeDelete Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daevtBeforeDeleteC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"daevtBeforeDeleteX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"daevtBeforeDeleteA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"daevtBeforeDeleteS"}

Occurs before a selected record is deleted in a **DBGrid** control.

Syntax

Private Sub *object*_**BeforeDelete (**[*index* **As Integer**,] *cancel* **As Integer)**

The BeforeDelete event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that identifies a control if it is in a control array.
cancel	A Boolean expression that determines whether a record is deleted, as described in Settings.

Settings

The settings for *cancel* are:

Setting	Description
True	Leaves focus on control and doesn't delete the record.
False	(Default) Continues with delete operation and enables change of focus.

Remarks

When the user selects a record selector in the control and presses DEL or CTL+X, the BeforeDelete event is triggered before the selected row is deleted.

Once the row is deleted, the AfterDelete event is triggered. The row selected for deletion is available in the collection provided by the **SelBookmarks** property.

If your event procedure sets the *cancel* argument to **True**, the row isn't deleted.

If more than one row is selected, the error message `Multiple rows cannot be deleted` is displayed.

BeforeDelete Event Example

This example displays a message that asks the user to confirm a deletion in a grid.

```
Private Sub DataGrid1_BeforeDelete (Cancel As Integer)
   Dim mResult As Integer
   mResult = MsgBox("Are you sure that you want to delete " &
DataGrid1.SeletedRows &  " record?", _
    vbYesNo And vbQuestion, "Delete Confirmation")
   If mResult = vbNo Then Cancel = True
End Sub
```

BeforeInsert Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daevtBeforeInsertC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"daevtBeforeInsertX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"daevtBeforeInsertA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"daevtBeforeInsertS"}

Occurs before new records are inserted into a **DBGrid** control.

Syntax

Private Sub *object*_**BeforeInsert (**[*index* **As Integer**,] *cancel* **As Integer)**

The BeforeInsert event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that identifies a control if it is in a control array.
cancel	A Boolean expression that determines if a record is added, as described in Settings.

Settings

The settings for *cancel* are:

Setting	Description
True	Leaves focus on control and doesn't add a new record
False	(Default) Continues with copy and enables change of focus

Remarks

When the user selects the new record (at the bottom of the **DBGrid** control) and enters a character in one of the cells, the BeforeInsert event is triggered, followed by the BeforeUpdate, AfterUpdate and AfterInsert events.

If your event procedure sets the *cancel* argument to **True**, the row isn't inserted and the cell is cleared.

When the BeforeInsert event is triggered, the record has not been added to the database. The new record exists in the **DBGrid** control's copy buffer until this event procedure ends.

After the AfterInsert event is finished, the new record row in the **DBGrid** control is reinitialized and the edited record becomes the last row in the **DBGrid** control.

BeforeInsert Event Example

This example displays a message that asks the user to confirm the addition of a new record.

```
Private Sub DataGrid1_BeforeInsert (Cancel As Integer)
   Dim mResult As Integer
   mResult = MsgBox("Confirm: Add a new record?", _
    vbYesNo And vbQuestion, "Confirmation")
   If mResult = vbNo Then Cancel = True
End Sub
```

BeforeUpdate Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daevtBeforeUpdateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"daevtBeforeUpdateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"daevtBeforeUpdateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"daevtBeforeUpdateS"}

Occurs before data is moved from a **DBGrid** control to the control's copy buffer.

Syntax

Private Sub *object*_**BeforeUpdate (**[*index* **As Integer**,] *cancel* **As Integer)**

The BeforeUpdate event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that identifies a control if it is in a control array.
cancel	A Boolean expression that determines if data is copied, as described in Settings.

Settings

The settings for *cancel* are:

Setting	Description
True	Leaves focus on control and doesn't copy data.
False	(Default) Continues with copy operation and enables change of focus.

Remarks

When the user moves to another row or the **Recordset** object's **Update** method is executed, data is moved from the **DBGrid** control's copy buffer to the **Data** control's copy buffer and written to the database.

Just before the data is moved from the **DBGrid** control's copy buffer back into the **Data** control's copy buffer, the BeforeUpdate event is triggered. Unless the copy operation is canceled, the AfterUpdate event is triggered after the data has been moved back into the **Data** control's copy buffer and written to the database. The updated record is available by using the **Bookmark** property of the **DBGrid** control.

If you set the BeforeUpdate event *cancel* argument to **True**, focus remains on the control, neither the AfterUpdate or LostFocus event is triggered, and the record isn't saved to the database.

The BeforeUpdate event occurs before the AfterUpdate and LostFocus events for this control, or before the GotFocus event for the next control in the tab order.

This event occurs even if the control isn't bound.

Unlike the Change event, changing data in a control or record using code doesn't trigger this event.

You can use this event to validate data in a bound control record before permitting the user to commit the change to the **Data** control's copy buffer. By setting the *cancel* argument to **True**, the user can't move focus from the control until the application determines whether the data can be safely moved back to the **Data** control's copy buffer.

BeforeUpdate Event Example

This example displays a message that tells the user to enter a value in the first column before the grid can be updated.

```
Private Sub DataGrid1_BeforeUpdate (Cancel As Integer)
   If DataGrid1.Columns(1).Value = "" Then
      MsgBox "You must enter value in the first column!"
      Cancel = True
   End If
End Sub
```

ColResize Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daevtColResizeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"daevtColResizeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"daevtColResizeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"daevtColResizeS"}

Occurs before the Paint event when a user resizes a column of a **DBGrid** control.

Syntax

Private Sub *object*_**ColResize (**[*index* **As Integer**,] *colindex* **As Integer**, *cancel* **As Integer)**

The ColResize event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that identifies a control if it is in a control array.
colindex	An integer that identifies the column.
cancel	A Boolean expression that determines whether a column is resized, as described in Settings.

Settings

The settings for *cancel* are:

Setting	Description
True	Cancels the change, restores column to its original width, and cancels the pending Paint event.
False	(Default) Continues with width change and subsequent Paint event.

Remarks

When the user resizes a column, the ColResize event is triggered. Your event procedure can accept the change, alter the degree of change, or cancel the change completely.

If you set the *cancel* argument to **True**, the column width is restored and no Paint event is triggered. To alter the degree of change, set the **Width** property of the **Column** object to the desired value.

Executing the **Refresh** method within the procedure causes the control to be repainted even if the *cancel* argument is **True**.

ColResize Event Example

This example resizes all the columns to the size of the first column if the user sizes the first column.

```
Private Sub DataGrid1_ColResize (ColIndex As Integer, Cancel As Integer)
   Dim nCol As Column
   If ColIndex = 1 Then
      For Each nCol In DataGrid1.Columns
         nCol.Width = DataGrid.Columns(1).Width
      Next
   End If
End Sub
```

HeadClick Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daevtHeadClickC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"daevtHeadClickX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"daevtHeadClickA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"daevtHeadClickS"}

Occurs when the user clicks on the header for a particular column of a **DBGrid** control.

Syntax

Private Sub *object*_**HeadClick (**[*index* **As Integer**,] *colindex* **As Integer)**

The HeadClick event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that identifies a control if it is in a control array.
colindex	An integer that identifies the column.

Remarks

One possible use for this event is to resort the **Recordset** object based on the selected column.

HeadClick Event Example

This example sorts the record source of the **Data** control based on which column the user clicked.

```
Private Sub DataGrid1_HeadClick (ColIndex As Integer)
   Data1.RecordSource = "Select * From Publishers Order By " & _
    DataGrid1.Columns(ColIndex).DataField
   Data1.Refresh
End Sub
```

RowResize Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daevtRowResizeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"daevtRowResizeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"daevtRowResizeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"daevtRowResizeS"}

Occurs before the Paint event when a user resizes a row in a **DBGrid** control.

Syntax

Private Sub *object*_**RowResize (**[*index* **As Integer**,] *cancel* **As Integer)**

The RowResize event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that identifies a control if it is in a control array.
cancel	A Boolean expression that determines if a change is canceled, as described in Settings.

Settings

The settings for *cancel* are:

Setting	Description
True	Cancels the change, restores row to its original height, and cancels the pending Paint event.
False	(Default) Continues with change of row height and subsequent Paint event.

Remarks

The user can resize the **DBGrid** control rows using the mouse. When the user changes the height, the RowResize event is triggered. Your event procedure can accept the change, alter the degree of change, or cancel the change completely.

The **DBGrid** control's **RowHeight** property determines the height for all rows in the control.

If you set the *cancel* argument to **True**, the row height is restored and no Paint event is fired. To alter the degree of change, set the **RowHeight** property to the desired value.

Executing the **Refresh** method within the procedure causes the control to be repainted even if the *cancel* argument is **True**.

RowResize Event Example

This example ensures that there are at least five visible rows in the grid.

```
Private Sub DataGrid1_RowResize (Cancel As Integer)
   If DataGrid1.VisibleRows < 5 Then Cancel = True
End Sub
```

UpdateControls Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"damthUpdateControlsC;daobjRecordset;daproEdit;daproValidationRule;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"damthUpdateControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"damthUpdateControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"damthUpdateControlsS"}

Gets the current record from a **Data** control's Recordset object and displays the appropriate data in controls bound to a **Data** control. Doesn't support named arguments.

Syntax

object.**UpdateControls**

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

Remarks

Use this method to restore the contents of bound controls to their original values, as when a user makes changes to data and then decides to cancel the changes.

This method creates the same effect as making the current record current again, except that no events occur.

The **UpdateControls** method terminates any pending **Edit** or **AddNew** operation.

UpdateRecord Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"damthUpdateRecordC;daobjRecordset;daproEdit;daproValidationRule;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"damthUpdateRecordX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"damthUpdateRecordA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"damthUpdateRecordS"}

Saves the current values of bound controls. Doesn't support named arguments.

Syntax

object.**UpdateRecord**

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

Remarks

Use this method to save the current contents of bound controls to the database during the Validate event without triggering the Validate event again. Using this method avoids creating a cascading event.

The **UpdateRecord** method has the same effect as executing the **Edit** method, changing a field, and then executing the **Update** method, except that no events occur.

You can use this method to avoid triggering the Validate event.

Whenever you attempt to update a record in the database, any validation rules must be satisfied before the record is written to the database. These rules are established by setting the **ValidationRule** property or, in the case of Microsoft SQL Server, by Transact SQL defaults, rules, and triggers written to enforce referential and data integrity.

In some cases, the update may not occur because the operation violates referential integrity constraints, the page containing the record is locked, the database or **Recordset** object isn't updatable, or the user doesn't have permission to perform the operation. Any of these conditions generates a trappable error.

Database Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"dacolFieldS;dacolIndex;dacolTableDefS;daobjDatabase;daproConnect;daproDatabaseC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"daproDatabaseX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"daproDatabaseA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"daproDatabaseS"}

Returns a reference to a **Data** control's underlying **Database** object.

Syntax

object.**Database**
Set *databaseobject* **=** *object*.**Database** (Professional and Enterprise Editions only)

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

Part	Description
databaseobject	An object expression that evaluates to an valid **Database** object created by the **Data** control.
object	An object expression that evaluates to an object in the Applies To list.

Remarks

The **Database** object created by the **Data** control is based on the control's **DatabaseName**, **Exclusive**, **ReadOnly**, and **Connect** properties.

Database objects have properties and methods you can use to manage your data. You can use any method of a **Database** object with the **Database** property of a **Data** control, such as **Close** and **Execute**. You can also examine the internal structure of the **Database** by using its **TableDefs** collection, and in turn, the **Fields** and **Indexes** collections of individual **TableDef** objects.

Although you can create a **Recordset** object and pass it to a **Data** control's **Recordset** property, you can't open a database and pass the newly created **Database** object to the **Data** control's **Database** property.

Data Type

Database

DatabaseName Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjDatabase;daproConnect;vbproBooksOnlineJumpTopic;vbproDatabaseNameC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"daproDatabaseX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDatabaseNameA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDatabaseNameS"}

Returns or sets the name and location of the source of data for a **Data** control.

Syntax

object.**DatabaseName** [**=** *pathname*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
pathname	A string expression that indicates the location of the database file(s) or the Data Source name for ODBC data sources.

Remarks

If your network system supports it, the *pathname* argument can be a fully qualified network path name such as \\Myserver\Myshare\Database.mdb.

The database type is indicated by the file or directory that *pathname* points to, as follows:

pathname Points To...	Database Type
.mdb file	Microsoft Access database
Directory containing .dbf file(s)	dBASE database
Directory containing .xls file	Microsoft Excel database
Directory containing .dbf files(s)	FoxPro database
Directory containing .wk1, .wk3, .wk4, or .wks file(s)	Lotus Database
Directory containing .pdx file(s)	Paradox database
Directory containing text format database files	Text format database

For ODBC databases, such as SQL Server and Oracle, this property can be left blank if the control's **Connect** property identifies a data source name (DSN) that identifies an ODBC data source entry in the registry.

If you change the **DatabaseName** property after the control's **Database** object is open, you must use the **Refresh** method to open the new database.

Note For better performance when accessing external databases, it's recommended that you attach external database tables to a Microsoft Jet engine database (.mdb) and use the name of the Jet .mdb database in the **DatabaseName** property.

Data Type

String

DataChanged Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDataChangedC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDataChangedX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDataChangedA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDataChangedS"}

Returns or sets a value indicating that the data in the bound control has been changed by some process other than that of retrieving data from the current record. Not available at design time.

Syntax

object.**DataChanged** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that indicates whether data has changed, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	The data currently in the control isn't the same as in the current record.
False	(Default) The data currently in the control, if any, is the same as the data in the current record.

Remarks

When a **Data** control moves from record to record, it passes data from fields in the current record to controls bound to the specific field or the entire record. As data is displayed in the bound controls, the **DataChanged** property is set to **False**. If the user or any other operation changes the value in the bound control, the **DataChanged** property is set to **True**. Simply moving to another record doesn't affect the **DataChanged** property.

When the **Data** control starts to move to a different record, the Validate event occurs. If **DataChanged** is **True** for any bound control, the **Data** control automatically invokes the **Edit** and **Update** methods to post the changes to the database.

If you don't wish to save changes from a bound control to the database, you can set the **DataChanged** property to **False** in the Validate event.

Inspect the value of the **DataChanged** property in your code for a control's Change event to avoid a cascading event. This applies to both bound and unbound controls.

Data Type

Integer (Boolean)

DataField Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daproDataFieldC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"daproDataFieldX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"daproDataFieldA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"daproDataFieldS"}

Returns or sets a value that binds a control to a field in the current record

Syntax

object.**DataField** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A string expression that evaluates to the name of one of the fields in the **Recordset** object specified by a **Data** control's **RecordSource** and **DatabaseName** properties.

Remarks

Bound controls provide access to specific data in your database. Bound controls that manage a single field typically display the value of a specific field in the current record. The **DataSource** property of a bound control specifies a valid **Data** control name, and the **DataField** property specifies a valid field name in the **Recordset** object created by the **Data** control. Together, these properties specify what data appears in the bound control.

When you use a **QueryDef** object or SQL statement that returns the results of an expression, the field name is automatically generated by the Microsoft Jet database engine. For example, when you code an SQL aggregate function or an expression in your SQL query, unless you alias the aggregate fields using an AS clause, the field names are automatically generated. Generally, the expression field name is Expr1 followed by a three-character number starting with 000. The first expression returned would be named Expr1000.

It's recommended that you code your SQL queries to alias expression columns as shown below:

```
Data1.RecordSource = "Select AVG(Sales)  " _
   & " AS AverageSales From SalesTable"
Text1.DataField = "AverageSales"
Data1.Refresh
```

Note Make sure the **DataField** property setting is valid for each bound control. If you change the setting of a **Data** control's **RecordSource** property and then use **Refresh**, the **Recordset** identifies the new object. This may invalidate the **DataField** settings of bound controls and produce a trappable error.

Data Type

String

DataSource Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDataSourceC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDataSourceX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDataSourceA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDataSourceS"}

Sets a value that specifies the **Data** control through which the current control is bound to a database. Not available at run time.

Remarks

To bind a control to a field in a database at run time, you must specify a **Data** control in the **DataSource** property at design time using the Properties window.

To complete the connection with a field in the **Recordset** managed by the **Data** control, you must also provide the name of a **Field** object in the **DataField** property. Unlike the **DataField** property, the **DataSource** property setting isn't available at run time.

Data Type

String

Exclusive Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daproConnect;vbproBooksOnlineJumpTopic;vbproExclusiveC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproExclusiveX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproExclusiveA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproExclusiveS"}

Returns or sets a value that indicates whether the underlying database for a **Data** control is opened for single-user or multi-user access.

Syntax

object.**Exclusive** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that determines user access, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	The database is open for single-user access. No one else can open the database until it's closed.
False	(Default) The database is open for multi-user access. Other users can open the database and have access to the data while it's open.

Remarks

The *value* of this property, along with the **DatabaseName**, **ReadOnly**, and **Connect** properties, is used to open a database. In the Professional and Enterprise Editions, this property corresponds to the *exclusive* argument in the **OpenDatabase** method.

The **Exclusive** property is used only when opening the **Database**. If you change the value of this property at run time, you must use the **Refresh** method for the change to take effect. If someone else already has the database open, you can't open it for exclusive use and a trappable error results.

Database operations are faster if the database is opened for exclusive use.

After you open a database for exclusive use, your application can have as many instances open as necessary. However, other applications running on your system are not permitted to open the database.

The **Exclusive** property is ignored for databases accessed through ODBC.

Data Type

Boolean

Options Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"damthOpenRecordset;vbproBooksOnlineJumpTopic;vbproOptionsC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOptionsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOptionsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOptionsS"}

Returns or sets a value that specifies one or more characteristics of the **Recordset** object in the control's **Recordset** property.

Syntax

object.**Options** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A constant or value that specifies a characteristic of a **Recordset**, as described in Settings.

Settings

Use one or more of the following values to set the **Options** property. If you use more than one option, you must add their values:

Constant	Value	Description
dbDenyWrite	1	In a multi-user environment, other users can't make changes to records in the **Recordset**.
dbDenyRead	2	In a multi-user environment, other users can't read records (table-type **Recordset** only).
dbReadOnly	4	You can't make changes to records in the **Recordset**.
dbAppendOnly	8	You can add new records to the **Recordset**, but you can't read existing records.
dbInconsistent	16	Updates can apply to all fields of the **Recordset**, even if they violate the join condition.
dbConsistent	32	(Default) Updates apply only to those fields that don't violate the join condition.
dbSQLPassThrough	64	When using **Data** controls with an SQL statement in the **RecordSource** property, sends the SQL statement to an ODBC database, such as a SQL Server or Oracle database, for processing.
dbForwardOnly	256	The Recordset object supports forward-only scrolling. The only move method allowed is **MoveNext**. This option cannot be used on **Recordset** objects manipulated with the **Data** control.

dbSeeChanges	512	Generate a trappable error if another user is changing data you are editing.

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

If you change the **Options** property at run time, you must use the **Refresh** method for the change to have any effect.

In the Professional and Enterprise Editions, this property corresponds to the *options* argument in the **OpenRecordset** method.

To set more than one value for this property, you can combine options by adding values together. For example, to set both **dbAppendOnly** and **dbInconsistent** you can use this code:

```
Data1.Options = dbAppendOnly + dbInconsistent
```

To determine if the property contains a specific value, you can use the **And** operator. For example, to find out if the **Recordset** is open for read-only access, you could use this code:

```
If Data1.Options And dbReadOnly Then...
```

Using both **dbInconsistent** and **dbConsistent** results in consistent updates, the default for **Recordset** objects.

Note The **dbSQLPassThrough** option can only be used when creating dynaset- or snapshot-type **Recordset** objects and is supported only to provide compatibility with previous versions. For better performance and functionality, you should use a previously created SQL PassThrough **QueryDef** object and set the **Data** control's **Recordset** property to a **Recordset** object created with the **QueryDef**.

Note If you attempt to access a SQL Server 6.0 table that includes an identity column, you can trigger an erroneous 3622 error. To prevent this problem, use the **dbSeeChanges** option with the **Options** property or **OpenRecordset** method.

Data Type

Integer

ReadOnly Property (Data Access)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"damthOpenDatabase;vbproBooksOnlineJumpTopic;vbproReadOnlyDAC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbProReadOnlyDAX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproReadOnlyDAA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbProReadOnlyDAS"}

Returns or sets a value that determines whether the control's Database is opened for read-only access.

Syntax

object.**ReadOnly** [**=** *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that determines read/write access, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	The control's **Database** object is opened with read-only access. Changes to data aren't allowed.
False	(Default) The control's **Database** is opened with read/write access to data.

Remarks

Use the **ReadOnly** property with a **Data** control to specify whether data in the underlying **Database** can be changed. For example, you might create an application that only displays data. Accessing a **Database** using read-only result sets is also faster.

For a **Data** control, this property is used only the first time a database is opened by your application. If your application subsequently opens other instances of the database, the property is ignored. For a change in this property to take effect, you must close all instances of the database and then use the **Refresh** method.

In the Professional and Enterprise Editions, this property corresponds to the *readonly* argument in the **OpenDatabase** method.

Data Type

Boolean

RecordSource Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"dacolTableDefS;daobjQueryDef;daproIndex;vbproBooksOnlineJumpTopic;vbproRecordSourceC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRecordSourceX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRecordSourceA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRecordSourceS"}

Returns or sets the underlying table, SQL statement, or **QueryDef** object for a **Data** control.

Syntax

object.**RecordSource** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A string expression specifying a name, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
A table name	The name of one of the tables defined in the **Database** object's **TableDefs** collection.
An SQL query	A valid SQL string using syntax appropriate for the data source.
A **QueryDef**	The name of one of the **QueryDef** objects in the **Database** object's **QueryDefs** collection — when accessing a Jet database.

Remarks

The **RecordSource** property specifies the source of the records accessible through bound controls on your form.

If you set the **RecordSource** property to the name of an existing table in the database, all of the fields in that table are visible to the bound controls attached to the **Data** control. For table-type recordsets (**RecordsetType = vbRSTypeTable**), the order of the records retrieved is set by the **Index** object that you select using the **Index** property of the **Recordset**. For dynaset-type and snapshot-type **Recordset** objects, you can order the records by using a SQL statement with an Order By clause in the **RecordSource** property of the **Data** control. Otherwise, the data is returned in no particular order.

If you set the **RecordSource** property to the name of an existing **QueryDef** in the database, all fields returned by the **QueryDef** are visible to the bound controls attached to the **Data** control. The order of the records retrieved is set by the **QueryDef** object's query. For example, the **QueryDef** may include an ORDER BY clause to change the order of the records returned by the **Recordset** created by the **Data** control or a WHERE clause to filter the records. If the **QueryDef** doesn't specify an order, the data is returned in no particular order.

Note At design-time, the **QueryDef** objects displayed in the Properties window for the **RecordSource** property are filtered out to display only **QueryDef** objects that are usable with the **Data** control. **QueryDef** objects which have parameters, and **QueryDef** objects which have the following types are not displayed: **dbQAction**, **dbQCrosstab**, **dbQSQLPassThrough** and **dbQSetOperation**.

If you set the **RecordSource** property to an SQL statement that returns records, all fields returned by

the SQL query are visible to the bound controls attached to the **Data** control. This statement may include an ORDER BY clause to change the order of the records returned by the **Recordset** created by the **Data** control or a WHERE clause to filter the records. If the database you specify in the **Database** and **Connect** property isn't a Microsoft Jet engine database, and if the **dbSQLPassThrough** option is set in the **Options** property, your SQL query must use the syntax required by that database engine.

Note Whenever your **QueryDef** or SQL statement returns a value from an expression, the field name of the expression is created automatically by the Microsoft Jet database engine. Generally, the name is Expr1 followed by a three-character number beginning with 000. For example, the first expression would be named: Expr1000.

In most cases you'll want to alias expressions so you know the name of the column to bind to the bound control. See the SQL SELECT statement AS clause for more information.

After changing the value of the **RecordSource** property at run time, you must use the **Refresh** method to enable the change and rebuild the **Recordset**.

At run time, if the **Recordset** specifies an invalid **Table** name, **QueryDef** name, or contains invalid SQL syntax, a trappable error will result. If this error occurs during the initial Form_Load procedure, the error is not trappable.

Note Make sure each bound control has a valid setting for its **DataField** property. If you change the setting of a **Data** control's **RecordSource** property and then use **Refresh**, the **Recordset** identifies the new object. This may invalidate the **DataField** settings of bound controls and cause a trappable error.

Data Type

String

Recordset Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjQueryDef;daproDataUpdatable;daproRecordCount;daproUpdatable;vbproBooksOnlineJumpTopic;vbproRecordsetC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRecordsetX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRecordsetA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRecordsetS"}

Returns or sets a **Recordset** object defined by a **Data** control's properties or by an existing **Recordset** object.

Syntax

Set *object*.**Recordset** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An object variable containing a **Recordset** object.

Remarks

The **Data** control is automatically initialized when your application starts before the initial Form_Load procedure. If the **Connect**, **DatabaseName**, **Options**, **RecordSource**, **Exclusive**, **ReadOnly** and **RecordsetType** properties are valid, or if you set these **Data** control properties at run time and use the **Refresh** method, the Microsoft Jet database engine attempts to create a new **Recordset** object based on those properties. This **Recordset** is accessible through the **Data** control's **Recordset** property. If, however, one or more of these properties is set incorrectly at design time, an untrappable error occurs when Visual Basic attempts to use the properties to open the specified database and create the **Recordset** object.

You can use the **Recordset** property as you would any other **Recordset** object. For example, you can use any of the **Recordset** methods or properties and examine the structure of the **Recordset** object's underlying schema.

You can also request the type of **Recordset** to be created by setting the **Data** control's **RecordsetType** property. If you don't request a specific type, a dynaset-type **Recordset** is created. Using the **RecordsetType** property, you can request to create either a table-, snapshot- or dynaset-type **Recordset**. However, if the Jet engine can't create the type requested, a trappable error occurs.

In many cases, the default type and configuration of the **Recordset** object created is extremely inefficient. That is, you might not need an updatable, fully-scrollable, keyset-type cursor to access your data. For example, a read-only, snapshot-type **Recordset** might be far faster to create than the default **Recordset**. Be sure to choose the most efficient Type, **Exclusive**, **Options** and **ReadOnly** properties possible for your situation.

The type of **Recordset** created can be determined at run time by examining the **Recordset** property's **Type** property or the **Data** control's **RecordsetType** property. Note, however, that the constants used for the type of **Recordset** created are different. For example:

```
If Data1.Recordset.Type = dbOpenDynaset Then ...
If Data1.RecordsetType = dbDynasetType Then ...
```

A **Recordset** might not be updatable even if you request a dynaset- or table-type **Recordset**. If the underlying database, table, or field isn't updatable, all or portions of your **Recordset** may be read-only. Examine the **Database** and **Recordset** objects' **Updatable** property or the **Field** object's **DataUpdatable** property to determine if your code can change the records. Even when the **DataUpdatable** property returns **True**, there are situations where the underlying data fields might *not* be updatable if, for example, you do not have sufficient permissions to make changes. Other factors

can also prevent fields from being updatable.

The number of records returned by the **Recordset** can be determined by moving to the last record in the **Recordset** and examining the **Recordset** object's **RecordCount** property. Before you move to the last record, the value returned by the **RecordCount** property only reflects the number of rows processed by the Jet engine. The following example shows how you can combine the **RecordCount** property of a **Recordset** with the **Recordset** property to display the number of records in a **Data** control's recordset:

```
Data1.Recordset.MoveLast
MsgBox "Records: " & Data1.Recordset.RecordCount
```

Professional and Enterprise Editions

If you create a **Recordset** object using either code or another **Data** control, you can set the **Recordset** property of the **Data** control to this new **Recordset**. Any existing **Recordset** in the **Data** control, and the **Database** object associated with it are released when a new **Recordset** is assigned to the **Recordset** property.

Note When the **Recordset** property is set, the **Data** control doesn't close the current **Recordset** or **Database,** but it does release it. If there are no other users, the database is closed automatically. You may wish to consider closing the **Recordset** and **Database** associated with the **Data** control prior to setting the **Recordset** property using the **Close** method.

Make sure the **DataField** properties of the bound controls connected to the **Data** control are set to match the new **Recordset** object's field names.

For example, to create a **Recordset** in code and pass it to an existing **Data** control:

```
Dim Db As Database, Rs As Recordset' Defined as public variables.
Sub ApplyRecordset()
   Set Db = Workspaces(0).OpenDatabase("BIBLIO.MDB")
   Set Rs = Db.OpenRecordset("AUTHORS") ' Defaults to Table object.
   Set Data1.Recordset = Rs  ' Assign Recordset.
   Data1.Recordset.Index = "PrimaryKey"
   Debug.print Rs.Type  ' Show type created.
End Sub
```

You can use this technique to create an MDI parent and child data connection with a single hidden **Data** control on the MDI parent form and another visible **Data** control on the MDI child. In the MDI child's Form_Load event, set the child's **Data** control **Recordset** property to the parent's **Data** control **Recordset** property. Using this technique synchronizes all the child forms and their bound controls with the parent.

Note The **Data** control doesn't support forward-only **Recordset** objects. If you try to assign a forward-only **Recordset** object to the **Recordset** property of the **Data** control, a trappable error results.

All **Recordset** objects created by the **Data** control are created in **Workspaces**(0) except ODBCDirect (**DefaultType** = **dbUseODBC**) **Recordset** objects. If you need to use the **Data** control to manipulate a database in another **Workspace**, use the technique shown above to open the database in the desired **Workspace**, create a new **Recordset** and set the **Data** control's **Recordset** property to this new **Recordset**.

Important You can always reference the properties of the **Data** control's **Recordset** by using the **Recordset** property. By directly referencing the **Recordset**, you can determine the **Index** to use with **Table** objects, the **Parameters** collection of a **QueryDef,** or the **Recordset** type.

Data Type

Recordset

BOFAction, EOFAction Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjRecordset;daproBOF;vbproBOFActionC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBOFActionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBOFActionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBOFActionS"}

Returns or sets a value indicating what action the **Data** control takes when the **BOF** or **EOF** properties are **True**.

Syntax

object.**BOFAction** [**=** *integer*]
object.**EOFAction** [**=** *integer*]

The **BOFAction** and **EOFAction** property syntax's have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list
integer	An integer value that specifies an action, as described in Settings

Settings

For the **BOFAction** property, the settings for *integer* are:

Setting	Value	Description
vbBOFActionMoveFirst	0	**MoveFirst** (Default): Keeps the first record as the current record.
vbBOFActionBOF	1	**BOF**: Moving past the beginning of a **Recordset** triggers the **Data** control Validate event on the first record, followed by a Reposition event on the invalid (**BOF**) record. At this point, the Move Previous button on the **Data** control is disabled.

For the **EOFAction** property, the settings for *integer* are:

Setting	Value	Description
vbEOFActionMoveLast	0	**MoveLast** (Default): Keeps the last record as the current record.
vbEOFActionEOF	1	**EOF**: Moving past the end of a **Recordset** triggers the **Data** control's Validation event on the last record, followed by a Reposition event on the invalid (**EOF**) record. At this point, the MoveNext button on the **Data** control is disabled.
vbEOFActionAddNew	2	**AddNew**: Moving past the last record triggers the **Data** control's Validation event to occur on the current record, followed by an automatic **AddNew**, followed by a Reposition event on the new record.

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

If you set the **EOFAction** property to **vbEOFActionAddNew**, once the user moves the current record pointer to **EOF** using the **Data** control, the current record is positioned to a new record in the copy buffer. At this point you can edit the newly added record. If you make changes to the new record and the user subsequently moves the current record pointer using the **Data** control, the record is automatically appended to the **Recordset**. If you don't make changes to this new record, and reposition the current record to another record, the new record is discarded. If you use the **Data** control to position to another record while positioned over this new record, another new record is created.

When you use code to manipulate **Recordsets** created with the **Data** control, the **EOFAction** property has no effect — it only takes effect when manipulating the **Data** control with the mouse.

In situations where the **Data** control **Recordset** is returned with no records, or after the last record has been deleted, using the **vbEOFActionAddNew** option for the **EOFAction** property greatly simplifies your code because a new record is always editable as the current record. If this option is not enabled, you are likely to trigger a "No current record" error.

Data Type

Integer

IntegralHeight Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daproIntegralHeightC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"daproIntegralHeightX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"daproIntegralHeightA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"daproIntegralHeightS"}

Returns or sets a value indicating if the control displays partial items. Read-only at run time.

Syntax

object.**IntegralHeight** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that determines whether the list is resized, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	(Default) The list resizes itself to display only complete items.
False	The list doesn't resize itself even if the item is too tall to display completely.

Data Type

Boolean

RecordsetType Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"daobjRecordset;vbproBooksOnlineJumpTopic;vbproRecordsetTypeC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRecordsetTypeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRecordsetTypeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":""}

Returns or sets a value indicating the type of **Recordset** object you want the **Data** control to create.

Syntax

object.**RecordsetType** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A constant or value that specifies a type of **Recordset**, as described in Settings.

Settings

The settings for *value* are:

Setting	Value	Description
vbRSTypeTable	0	A table-type **Recordset**
vbRSTypeDynaset	1	(Default) A dynaset-type **Recordset**
vbRSTypeSnapshot	2	A snapshot-type **Recordset**

Remarks

If the Microsoft Jet database engine can't create the type of **Recordset** you requested, a trappable error occurs.

If you don't specify a **RecordsetType** before the **Data** control creates the **Recordset**, a dynaset-type **Recordset** is created (if possible).

If you create a **Recordset** without using the **Data** control (even with another **Data** control) and set the **Recordset** property with this new **Recordset** object, the **RecordsetType** property of the **Data** control is set to the **Recordset**.**Type** property of the new **Recordset**.

Important The **RecordsetType** property *value* doesn't correspond to the value used to identify **Recordset** object types. See the **OpenRecordset** method or the **Type** property for details.

In many cases, the default type and configuration of the **Recordset** object created is extremely inefficient. That is, you might not need an updatable, fully-scrollable, keyset-type cursor to access your data. For example, a read-only, forward-only, snapshot-type **Recordset** might be far faster to create than the default cursor. Be sure to choose the most efficient settings for the **RecordsetType**, **Exclusive**, **Options** and **ReadOnly** properties for your situation.

Data Type

Integer

DefaultType Property (Data Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"damthCreateWorkspace;daobjWorkspace;vbproBooksOnlineJumpTopic;vbproDefaultTypePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDefaultTypePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDefaultTypePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDefaultTypePropertyS"}

Returns or sets a value which determines the type of data source (Jet or ODBCDirect) that is used by the **Data** control.

Syntax

object.**DefaultType** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer constant or value that specifies the type of data source, as described in Settings.

Settings

The settings for *value* are:

Setting	Value	Description
dbUseODBC	1	Use ODBCDirect to access your data.
dbUseJet	2	(Default) Use the Microsoft Jet database engine to access your data.

Remarks

Setting the **DefaultType** property tells the **Data** control what type of data source (Jet or ODBCDirect) to use when creating a **Recordset**. The **DefaultType** property also determines the type of the underlying **Workspace** object used with the **Data** control. The Jet database engine will not be loaded unless this property is set to **dbUseJet**.

When setting the **DefaultType** property to **dbUseODBC**, Visual Basic creates a new **Workspace** object and adds it to the **Workspaces** collection. The **DefaultType** property of the **Data** control is similar to the *type* parameter of the **CreateWorkspace** method. When using **dbUseJet**, the default **Workspace** object is used.

Note When you select **dbUseODBC** for the **DefaultType** property, DAO routes all data access operations through a Remote Data Objects (RDO) DLL.

Choosing a Data Source

Data Access Objects (DAO) can be programmed to connect to remote ODBC data sources in one of two ways: through the Jet database engine or through Remote Data Objects (RDO) which bypasses Jet completely. Depending on the features and performance you need, either approach might make sense for your particular application.

Using ODBCDirect: This approach permits you to use the **Data** control against remote ODBC data sources by routing all DAO operations through the RDO interface. That is, when you establish a connection and create a **Recordset** object using the **Data** control, the Jet database engine is not loaded or used in any way. This also means that many of the DAO features provided by the Jet engine are not available on this **Workspace**. For example, you cannot perform heterogeneous joins, or access ISAM on .mdb databases without use of additional ODBC drivers. However, when you choose ODBCDirect, many RDO features not ordinarily supported by Jet are enabled.

Using The Jet Database Engine: Unless you enable ODBCDirect, the Jet database engine is loaded and performs all local and remote database operations. Once a Jet **Workspace** is created, it cannot be used to pass data to an ODBCDirect **Workspace**.

Data Type

Integer

DefaultCursorType Property (Data Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"damthOpenConnection;daobjWorkspace;vbproBooksOnlineJumpTopic;vbproDefaultCursorTypePropertyC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDefaultCursorTypePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDefaultCursorTypePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDefaultCursorTypePropertyS"}

Controls what type of cursor driver is used on the connection (ODBCDirect only) created by the **Data** control.

Syntax

object.**DefaultCursorType** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer constant or value that specifies a type of cursor driver , as described in Settings.

Settings

The settings for *value* are:

Setting	Value	Description
vbUseDefaultCursor	0	Let the ODBC driver determine which type of cursors to use.
vbUseODBCCursor	1	Use the ODBC cursor library. This option gives better performance for small result sets, but degrades quickly for larger result sets.
vbUseServerSideCursor	2	Use server side cursors. For most large operations this gives better performance, but might cause more network traffic.

Remarks

Use this property when the **DefaultType** property of the **Data** control is set to **dbUseODBC**. Refer to the **DefaultCursorDriver** property of the **Workspace** object for more information.

Data Type

Integer

ODBCDirect

A technology that allows you to access ODBC data sources through RDO by using DAO features that bypass the Microsoft Jet database engine.

When an ODBC data source is referenced in this manner, it is called an "ODBCDirect" data source. This is to distinguish it from an ODBC data source that is connected indirectly through the Jet Database Engine, which is called a "Jet-connected ODBC data source." The technology used in accessing the data source determines which DAO objects, methods, and properties can be used.

Recordset Property Example

This example uses a **Data** control to create a **Recordset** object containing all Titles published in 1994. The **Recordset** object created is accessible by referencing the **Recordset** property of the **Data** control.

```
Dim Rs As Recordset
Data1.DatabaseName = "BIBLIO.MDB"
Data1.RecordSource =
   "Select * From TITLES where [Year Published] = 1994"
Data1.Refresh
Set Rs = Data1.Recordset
If Rs.RecordCount > 0 Then
   Do Until Rs.EOF
      Debug.Print Rs!Title
      Rs.MoveNext
   Loop
Else
   MsgBox "No titles in 1994."
End If
```

This next example creates the same set of records as above in a **Recordset** object and assigns it to the **Data** control's **Recordset** property. To use this example, place a **Textbox**, **Commandbutton**, and **Data** control on a form and paste in the code.

```
Private Sub Command1_Click()
Dim DB As Database
Dim Rs As Recordset

Set DB = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")

Set Rs = DB.OpenRecordset
("Select * From TITLES where [Year Published] = 1994")

Set Data1.Recordset = Rs
Text1.DataField = "Title"
Data1.Refresh ' Force Data control to update now

End Sub
```

Exclusive Property Example

This example first opens a **Database** as single-user (exclusive) then changes it to multiuser access (non-exclusive).

```
Data1.DatabaseName = "BIBLIO.MDB"
Data1.RecordSource = "Publishers"
Data1.Exclusive = True
Data1.Refresh
...
Data1.Exclusive = False
Data1.Refresh
```

Database, DatabaseName Properties Example

This example examines the **Database** property of a data control and prints the name of each **Table** in the Debug window.

```
Sub PrintTableNames ()
   Dim Td As TableDef
   ' Set database file.
   Data1.DatabaseName = "BIBLIO.MDB"
   Data1.Refresh  ' Open the Database.
   ' Read and print the name of each table in the database.
   For Each Td in Data1.Database.TableDefs
      Debug.Print Td.Name
   Next
End Sub
```

LastModified Property Example

This example might be used with a menu in a program to enable users to return to the last record that was modified.

```
Data1.Recordset.Bookmark = Data1.Recordset.LastModified
```

DataChanged Property and Validate Event Example

This example illustrates simple data validation. In the Authors table in the Biblio.mdb database, there are two fields: Au_ID and Author. Since the value in Au_ID is used to uniquely identify the author, this value should not change. The example doesn't allow changes to the Au_ID field, which is bound to `Text1`.

```
Private Sub Data1_Validate (Action As Integer, Save As Integer)
   If Text1.DataChanged Then ' Check for change in data.
      MsgBox "You can't change the ID number."
      Text1.DataChanged = False    ' Don't save changed data.
   End If
   ...
End Sub
```

RecordsetType Property Example

This example uses the **Data** control to create a **Recordset** object and examines the **Data** control's **RecordsetType** property to determine what type of recordset was created.

```
Sub DisplayRecordsetType()
' Indicate type of Recordset wanted.
Data1.RecordsetType = vbRSTypeDynaset
Data1.DatabaseName = "BIBLIO.MDB"
Data1.RecordSource = "Authors"
Data1.Refresh

Select Case Data1.RecordsetType
   Case vbRSTypeTable
      Debug.print "Table-type Recordset created."
   Case vbRSTypeDynaset
      Debug.print "Dynaset-type Recordset created."
   Case vbRSTypeSnapshot
      Debug.print "Snapshot-type Recordset created."
End Select
End Sub
```

The Automation Manager could not be started. 'msgtext'

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

The Automation Manager could not be started on this computer for the reason given in the system error message appended at 'msgtext.'

For example, an `Out of Memory` error would prevent the Automation Manager from starting.

The Automation Manager was started with the following network protocols: 'protocols'.

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

Startup message (Information). The Automation Manager attempts to start using all the RPC network server protocols currently installed on the computer. A list of the protocols that were used successfully appears at 'protocols.'

You can view the list of RPC server protocols on a computer by accessing the following Windows Registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Rpc\ServerProtocols

The Automation Manager was unable to use network protocol 'protocol'. 'msgtext'

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

Startup message (Warning). The Automation Manager attempts to start with all the RPC network server protocols currently installed on the computer. The Automation Manager was unable to use the protocol 'protocol' for the reason given in the system error message appended at 'msgtext.'

You can view the list of RPC server protocols on a computer by accessing the following Windows Registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Rpc\ServerProtocols

The command line parameter 'parameter' was not recognized.

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

Startup message (Warning). The Automation Manager was started with an invalid command-line parameter, which was ignored. Valid parameters are:

Parameter	Description
/REGSERVER	Registers the Automation Manager in the Windows Registry.
/UNREGSERVER	Unregisters the Automation Manager.
/AUTOMATION	Accepted but ignored.
/EMBEDDING	Accepted but ignored.
/HIDDEN	Starts the Automation Manager without any visible window, so that it runs invisibly. This is particularly useful when the Automation Manager runs on a Win32 workstation. If the Automation Manager is started with this parameter there is no way for the user to close it.

The file 'file' could not be loaded. 'msgtext'

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

This error occurs during Setup. The Automation Manager has a dependency on the file named in the message, and that file cannot be found on the computer where Setup is being run, for the reason appended at 'msgtext'.

For example, Automation Manager depends on AUTPRX32.DLL.

The network protocol 'protocol' was needed but was not available. 'msgtext'

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

The Automation Manager was unable to use the indicated network protocol for the reason appended at 'msgtext.'

This error can occur when a client computer passes a reference to an object provided by an ActiveX component on the client computer, or on another network computer. (The client computer must itself be capable of running the Automation Manager for the first scenario to occur.)

The Automation Manager attempts to connect to the object using the same protocol that was used to pass the reference. If this protocol is not available on the network computer where the Automation Manager is running, the error occurs.

For example, the Named Pipes protocol (ncacn_np) is supported as an RPC client protocol under Windows 95, but not as an RPC server protocol. You can view the lists of RPC client and server protocols on a computer by accessing the following Windows Registry keys:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Rpc\ClientProtocols

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Rpc\ServerProtocols

On the client computer, the call that attempted to pass the object reference fails with the error &H800706D0, RPC_S_PROTSEQ_NOT_FOUND.

The preference setting 'setting' had an invalid value. The default setting will be used instead.

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

Automation Manager's RemoteActivationPolicy setting was invalid. The default setting, CreateIfKey (2), will be used. Valid values are:

Setting	Description
0	CreateNone. Do not allow creation of any CLSID.
1	CreateAll. Allow creation of any CLSID. Not recommended.
2	CreateIfKey. Allow creation of only those CLSIDs that include the subkey AllowRemoteActivation=Y
3	CreateIfAcl. Allow creation of a CLSID only if the user making the request has KEY_QUERY_VALUE permission on the CLSID key.

The location of the RemoteActivationPolicy preference is:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Automation Manager\RemoteActivationPolicy

There was a timeout processing a call to this machine. 'msgtext'

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

A call to a method of an object on this machine has timed out for the reason appended at 'msgtext.'

The timeout occurs if the client computer waits more than a specified number of milliseconds for an object, because another client computer is making a call to the same object.

This error only occurs when the Automation Manager serializes requests from two client computers that are accessing the same object. It does not occur when two client computers have separate objects of the same type, nor when OLE serializes requests to an out-of-process ActiveX component that is single-threaded.

The number of milliseconds for the timeout is specified in the Automation Manager's CallTimeout preference setting:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Automation Manager\CallTimeout

The default value is an unsigned four-byte integer, 0xFFFFFFFF, which represents over a hundred years, and thus is effectively infinite. Changing this preference affects all calls to methods of objects on this computer, hence changing it is not recommended.

There was an error accessing preferences in the registry. 'msgtext'

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

Automation Manager was unable to access its preferences in the Windows Registry for the reason appended at 'msgtext.'

For example, an error will occur if the Automation Manager is executed from a login that does not have read permission to the subkeys in the Windows Registry that contain Automation Manager's preferences:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Automation Manager\CallTimeout

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Automation Manager\RemoteActivationPolicy

There was an error adding information to the registry. 'msgtext'

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

Setup for the Automation Manager was unable to add the necessary entries to the Windows Registry for the reason appended at 'msgtext.' For example, an error will occur if Setup is executed from a login that does not have write access to the Windows Registry.

There was an error connecting to an object of type 'type'. 'msgtext'

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

A client computer has a reference to an object on this computer, and has attempted to pass that reference to a third computer. The third computer was unable to connect to the object for the reason appended at 'msgtext.'

'Type' is the fully qualified class name (programmatic ID), for example `Customer.Order`.

There was an error creating an object of type 'type' for 'user'. 'msgtext'

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

A client computer requested creation of an object of the specified type, for example by using the **CreateObject** function, or the **Set** statement with the **New** operator. The request failed for the reason given in the appended 'msgtext.'

If the class the object belongs to is in the Windows Registry of the network computer the Automation Manager is running on, 'type' is the fully qualified class name (programmatic ID), for example `Customer.Order`.

If the object type is not registered, for example when the ActiveX component that supplies it has not been installed on the network computer, 'type' is the CLSID of the object, as it appears in the Windows Registry of the client computer.

The user name associated with the request to create the object is inserted at 'user.'

This form of the error occurs only if both of the following are true:

- The Automation Manager must be running under the Microsoft Windows NT operating system
- The client computer must be using an Authentication level other than None (1).

Otherwise, the Automation Manager cannot determine the user name, and the form of the error without the user information is used.

There was an error creating an object of type 'type'. 'msgtext'

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

A client computer requested creation of an object of the specified type, for example by using the **CreateObject** function, or the **Set** statement with the **New** operator. The request failed for the reason given in the appended 'msgtext.'

If the class the object belongs to is in the Windows Registry of the network computer the Automation Manager is running on, 'type' is the fully qualified class name (programmatic ID).

For example, suppose the Customer server is installed on the network computer, the Automation Manager is using **CreateIfKey** security, and the CLSID entry for the Order class does not have the subkey AllowRemoteActivation=Y. The error message refers to the object type `Customer.Order`.

If the object type is not registered, for example when the ActiveX component that supplies it has not been installed on the network computer, 'type' is the CLSID of the object, as it appears in the Windows Registry of the client computer.

There was an error processing a call to this machine. 'msgtext'

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

A call to a method of an object provided by an ActiveX component running on this computer has failed for the reason appended at 'msgtext.'

This error occurs when the Automation Manager has received the method call from the client computer, but is unable to make the cross-process call to the component, or made the cross-process call but failed to receive the result.

This error occurs only for cross-process communication failures. It does not occur for errors raised in a component, or for COM or OLE exceptions; such errors are simply returned to the client application.

There was an error removing information from the registry. 'msgtext'

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

The Automation Manager was run with the **/UNREGSERVER** parameter, and was unable to remove the necessary entries from the Windows Registry for the reason appended at 'msgtext.' For example, an error will occur if the Automation Manager is executed in this fashion from a login that does not have write access to the Windows Registry.

BorderStyle Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproBorderStyleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBorderStyleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBorderStyleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBorderStyleOS"}

Returns or sets the border style for an object.

Syntax

object.**BorderStyle** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A value or constant that determines the border style, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
ccNone	0	(Default) No border or border-related elements.
ccFixedSingle	1	Fixed single.

Note The cc prefix refers to the Windows 95 controls. For the other controls, prefixes for the settings change with the specific control or group of controls. However, the description remains the same unless indicated.

Remarks

Setting **BorderStyle** for a **ProgressBar** control decreases the size of the chunks the control displays.

Image Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproImageActiveXControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproImageActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproImageActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproImageActiveXControlsS"}

Returns or sets a value that specifies which **ListImage** object in an **ImageList** control to use with another object.

Syntax

object.**Image** [= *index*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer or unique string specifying the **ListImage** object to use with *object*. The integer is the value of the **Index** property; the string is the value of the **Key** property.

Remarks

Before setting the **Image** property, you must associate an **ImageList** control with a **Toolbar**, **TreeView**, or **TabStrip** control by setting each control's **ImageList** property to an **ImageList** control.

At design time, put an **ImageList** control on the form and load it with images, each of which is a **ListImage** object assigned an index number in a **ListImages** collection. On the General tab in the control's Property Pages dialog box, select the **ImageList** you want from the **ImageList** list box, such as `ImageList1`. For **Tab** and **Button** objects, you can also specify the image you want to associate with these objects by typing the index number of the specific **ListImage** object in the Image field on the Tabs or Buttons tab.

At run time, use code like the following to associate an **ImageList** to a control and then a **ListImage** to a specific object:

```
Set TabStrip1.ImageList=ImageList1
TabStrip1.Tabs(1).Image=2
```

Use the **Key** property to specify an **ImageList** control's **ListImage** object when you wish your code to be self-documenting, as follows:

```
' Assuming there is a ListImage object with the Key property value =
' "close," use that image for a Toolbar button.
Toolbar1.Buttons(1).Image = "close"

' This is easier to read than just specifying an Index value, as below:
Toolbar1.Buttons(1).Image = 4 ' Requires that the ListImage object
' with Index property = 4 is the "close" image.
```

The value of the **Index** property of an object can change when objects in the collection are reordered, such as when you set the **Sorted** property to **True**. If you expect the **Index** property to change dynamically, it may be more useful to refer to objects in a collection by using the **Key** property.

If there are no images for a **Tabs** collection, the value of *index* is -1.

Index Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproIndexC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproIndexCustomX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproIndexCustomA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproIndexCustomS"}

Returns or sets the number that uniquely identifies an object in a collection.

Syntax

object.**Index**

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

Remarks

The **Index** property is set by default to the order of the creation of objects in a collection. The index for the first object in a collection will always be one.

The value of the **Index** property of an object can change when objects in the collection are reordered, such as when you set the **Sorted** property to **True**. If you expect the **Index** property to change dynamically, it may be more useful to refer to objects in a collection by using the **Key** property.

Key Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproKeyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproKeyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproKeyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproKeyS"}

Returns or sets a string that uniquely identifies a member in a collection.

Syntax

object.**Key** [= *string*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	A unique string identifying a member in a collection.

Remarks

If the string is not unique, an error will occur.

You can set the **Key** property when you use the **Add** method to add an object to a collection.

The value of the **Index** property of an object can change when objects in the collection are reordered, such as when you set the **Sorted** property to **True**. If you expect the **Index** property to change dynamically, refer to objects in a collection using the **Key** property.

In addition, you can use the **Key** property to make your Visual Basic project "self-documenting" by assigning meaningful names to the objects in a collection.

ShowTips Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTooltipsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTooltipsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTooltipsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTooltipsS"}

Returns or sets a value that determines whether ToolTips are displayed for an object.

Syntax

object.**ShowTips** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression specifying whether ToolTips are displayed, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	(Default) Each object in the control may display an associated string, which is the setting of the **ToolTipText** property, in a small rectangle below the object. This ToolTip appears when the user's cursor hovers over the object at run time for about one second.
False	An object will not display a ToolTip at run time .

Remarks

At design time you can set the **ShowTips** property on the General tab in the control's Property Pages dialog box.

ImageList Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproImageListC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproImageListX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproImageListA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproImageListS"}

Returns or sets the **ImageList** control, if any, that is associated with another control.

Syntax

object.**ImageList** [= *imagelist*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
imagelist	An object reference that specifies which **ImageList** control to use.

Remarks

For a control to use the **ImageList** property, you must place an **ImageList** control on the form. Then, at design time, you can set the **ImageList** property in the associated control's Property Pages dialog box. To associate an **ImageList** with a control at run time, set the control's **ImageList** property to the **ImageList** control you want to use, as in this example:

```
Set TabStrip1.ImageList = ImageList1
```

Clear Method (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthClearObjectC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthClearMethodX;vbmthClearObjectsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthClearObjectsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthClearObjectsS"}

Removes all objects in a collection.

Syntax

*object.***Clear**

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

Remarks

To remove only one object from a collection, use the **Remove** method.

Clear Method (ActiveX Controls) Example

This example adds six **Panel** objects to a **StatusBar** control, creating a total of seven **Panel** objects. A click on the form clears all **Panel** objects when their number reaches seven. If the number of **Panel** objects is less than seven, each click on the form will add a new **Panel** object to the control until the number seven is once again reached. To try the example, place a **StatusBar** control on a form and paste the code into the Declarations section. Run the example and click on the form to clear all **Panel** objects and subsequently add **Panel** objects.

```
Private Sub Form_Load()
   Dim pnlX As Panel ' Declare object variable for Panel objects.
   Dim I As Integer

   ' Add 6 Panel objects to the single default Panel object,
   ' making 7 Panel objects.
   For I = 1 to 6
      Set pnlX = StatusBar1.Panels.Add
   Next I
End Sub

Private Sub Form_Click()
   ' If the Count of the collection is 7, then clear the collection.
   ' Otherwise, add one Panel and use the collection's Count property
   ' to set its Style.
   If StatusBar1.Panels.Count = 7 Then
      StatusBar1.Panels.Clear
   Else
      Dim pnlX As Panel
      Set pnlX = StatusBar1.Panels.Add( , , "simple", 0)
      ' The Style property is enumerated from 0 to 6. Use the Panels
      ' Count property -1 to set the Style property for the new Panel.
      ' Display all panels regardless of form width.
      pnlX.minwidth = TextWidth("simple")
      pnlX.AutoSize = sbrSpring
      pnlX.Style = Statusbar1.Panels.Count - 1
   End If
End Sub
```

Remove Method (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthRemoveC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthRemoveX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthRemoveA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthRemoveS"}

Removes a specific member from a collection.

Syntax

object.**Remove** *index*

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer or string that uniquely identifies the object in the collection. An integer specifies the value of the **Index** property; a string specifies the value of the **Key** property.

Remarks

To remove all the members of a collection, use the **Clear** method.

Remove Method (ActiveX Controls) Example

This example adds six **Panel** objects to a **StatusBar** control, creating a total of seven **Panel** objects. When you click on the form, the code checks to see how many **Panel** objects there are. If there is only one **Panel** object, the code adds six **Panel** objects. Otherwise, it removes the first panel. To try the example, place a **StatusBar** control on a form and paste the code into the Declarations section. Run the example and click on the form to remove one **Panel** object at a time, and subsequently add **Panel** objects.

```
Private Sub Form_Load()
   Dim pnlX As Panel    ' Declare object variable for Panel objects.
   Dim i As Integer

   ' Add 6 Panel objects to the single default Panel object,
   ' making 7 Panel objects.
   For i = 1 To 6
      Set pnlX = StatusBar1.Panels.Add(, , , i)
      pnlX.AutoSize = sbrSpring
   Next i
End Sub

Private Sub Form_Click()
   ' If the Count of the collection is 1, add 6 Panel objects.
   ' Otherwise, remove the first panel from the collection.
   If StatusBar1.Panels.Count = 1 Then
      Dim sbrX As Panel
      Dim i As Integer
      For i = 1 To 6 ' Each panel has its style set by i.
         Set sbrX = StatusBar1.Panels.Add(, , , i)
         sbrX.AutoSize = sbrSpring
      Next i
   Else ' Remove the first panel.
      StatusBar1.Panels.Remove 1
   End If
End Sub
```

Value Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproValueActiveXControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproValueActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproValueActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproValueActiveXControlsS"}

Returns or sets the value of an object.

Syntax

object.**Value** [= *integer*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
integer	For a **Slider** control, a long integer that specifies the current position of the slider. For the **ProgressBar** control, an integer that specifies the value of the **ProgressBar** control. For other controls, see Settings below.

Settings

For the Button object, the settings for *integer* are:

Constant	Value	Description
tbrPressed	0	(Default). The button is not currently pressed or checked.
tbrUnpressed	1	The button is currently pressed or checked.

For the **3D Check Box**, **3D Command Button**, and **3D Group Push Button** controls, the settings for *integer* are:

Value	Description
True	The button is pressed.
False	(Default). The button is not pressed.

For the **3D Option Button** control, the settings for *integer* are:

Value	Description
True	The button is selected.
False	(Default). The button is not selected.

Remarks

- **Slider** control—returns or sets the current position of the slider. **Value** is always between the values for the **Max** and **Min** properties, inclusive, for a **Slider** control.
- **ProgressBar** control—returns or sets a value indicating an operation's approximate progress toward completion. Incrementing the **Value** property doesn't change the appearance of the **ProgressBar** control by the exact value of the **Value** property. **Value** is always in the range between the values for the **Max** and **Min** properties, inclusive. Not available at design time.
- **3D Command Button** control—returns or sets a value indicating whether the button is chosen; not available at design time. Setting the **Value** property to **True** in code invokes the button's Click event.

Value Property Example

This example uses the **Value** property to determine which icon from an associated **ImageList** control is displayed on the **Toolbar** control. To try the example, place a **Toolbar** control on a form and paste the code into the form's Declarations section. Then run the example.

```
Private Sub Toolbar1_ButtonClick(ByVal Button As Button)
   ' Use the Key value to determine which button has been clicked.
   Select Case Button.Key

   Case "Done" ' A check button.
      If Button.Value = vbUnchecked Then ' The button is unchecked.
         Button.Value = vbChecked  ' Check the button.
         ' Assuming there is a ListImage object with key "down."
         Button.Image = "down"
      Else
         Button.Value = vbUnchecked      ' Uncheck the button
         ' Assuming there is a ListImage object with key " up."
         Button.Image = "up"
      End If

   ' More Cases are possible.
   End Select
End Sub
```

HideSelection Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHideSelectionActiveXControlsC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHideSelectionActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHideSelectionActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHideSelectionActiveXControlsS"}

Returns or sets a value that specifies whether the selected item remains highlighted when a control loses focus.

Syntax

*object.***HideSelection** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying how a control is displayed when it loses focus, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) The items in the control are no longer selected when the control loses the focus.
False	The items remain selected after the control loses focus.

Remarks

Normally, the selected items in a control are hidden when the control loses focus. This is the default action of the property.

If you want the selected items to remain selected after the control loses focus, set the **HideSelection** property to **False**.

Text Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTextActiveXControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTextActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTextActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTextActiveXControlsS"}

Returns or sets the text contained in an object.

Syntax

object.**Text** [= *string*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	A string expression specifying the text appearing in the object.

Text Property (ActiveX Controls) Example

This example populates a **TreeView** control with the titles of files in a **ListBox** control. When an item in the **TreeView** control is clicked, the **Text** property is displayed in a **Label** on the form. To try the example, place **TreeView**, **Label**, and **ListBox** controls on a form and paste the code into the form's Declarations section. Run the example and click on any item to see its **Text** property.

```
Private Sub Form_Load()
   Dim nodX As Node  ' Declare an object variable for the Node.
   Dim i As Integer  ' Declare a variable for use as a counter.

   ' Add one Node to the TreeView control, and call it the first node
   Set nodX = TreeView1.Nodes.Add()
   nodX.Text = "First Node"

   'Populate the ListBox
      List1.AddItem "Node1"   ' Add each item to list.
      List1.AddItem "Node2"
      List1.AddItem "Node3"
      List1.AddItem "Node4"
      List1.AddItem "Node5"
      List1.AddItem "Node6"
      List1.AddItem "Node7"

   ' Add child nodes to the first Node object. Use the
   ' ListBox to populate the control.
   For i = 0 To List1.ListCount - 1
      Set nodX = TreeView1.Nodes.Add(1, tvwChild)
      nodX.Text = List1.List(i)
   Next i
   Treeview1.Nodes(1).Selected = True
   nodX.EnsureVisible   ' Make sure the node is visible.
End Sub

Private Sub TreeView1_NodeClick(ByVal Node As Node)
   ' Display the clicked Node object's Text property.
   Label1.Caption = Node.Text
End Sub
```

Windows 95 Control Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbidxWindows95ControlConstantsC;vbproBooksOnlineJumpTopic"}

The following constants are recognized by the ActiveX controls. As a result, they can be used anywhere in your code in place of the actual values.

- **BorderStyle Constants**
- **MousePointer Constants**

Use the Object Browser to view the intrinsic constants you can use with methods and properties. From the View menu, choose Object Browser, select the appropriate control library, and then the Constants object. You can scroll through the constants that appear under Methods/Properties.

Note Prefixes for the constants change with the specific control or group of controls. However, the description remains the same unless indicated.

MousePointer Constants

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

Constant	Value	Description
ccDefault	0	(Default) Shape determined by the object.
ccArrow	1	Arrow.
ccCross	2	Cross (cross-hair pointer).
ccIbeam	3	I Beam.
ccIcon	4	Icon (small square within a square).
ccSize	5	Size (four-pointed arrow pointing north, south, east, and west).
ccSizeNESW	6	Size NE SW (double arrow pointing northeast and southwest).
ccSizeNS	7	Size N S (double arrow pointing north and south).
ccSizeNWSE	8	Size NW, SE.
ccSizeEW	9	Size E W (double arrow pointing east and west).
ccUpArrow	10	Up Arrow.
ccHourglass	11	Hourglass (wait).
ccNoDrop	12	No Drop.
ccArrowHourglass	13	Arrow and hourglass.
cc ArrowQuestion	14	Arrow and question mark.
ccSizeAll	15	Size all.
ccCustom	99	Custom icon specified by the **MouseIcon** property.

Note The cc prefix refers to the custom controls. Prefixes for the constants change with the specific control or group of controls. However, the description remains the same unless indicated.

BorderStyle Constants

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

Constant	Value	Description
ccNone	0	(Default) No border or border-related elements.
ccFixedSingle	1	(Default for **ListView** control) Fixed single. There is a single line border around the control.

Note The cc prefix refers to the custom controls. The prefixes for the constants change with the specific control or group of controls. However, the description remains the same unless indicated.

Visual Basic Menus

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamnuContextMenusC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamnuContextMenusS"}

Visual Basic has two types of menus, built-in and shortcut.

Built-in Menus

Built-in menus appear on the menu bar across the top of the Visual Basic window. Each menu contains commands that relate to the menu name. For example, the Format menu contains commands used for formatting your form. Some of the commands have submenus that contains more specific commands. For example, the Toolbars command on the View menu has a submenu that contains the names of the toolbars and the Customize command. You can use the Customize command to modify the built-in menus or to add commands to the menu bar.

Shortcut Menus

Shortcut menus are menus containing frequently used commands that appear when you click the right mouse button or press SHIFT+F10.

Note To find information on a command on a menu, use the Search for Help On command on the Help menu and search for the name of the command.

Definition Command (Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdGoToDefinitionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdGoToDefinitionS"}

Displays the location in the Code window where the variable or procedure under the pointer is defined. If the definition is in a referenced library, it is displayed in the Object Browser.

Show Hidden Members Command (Object Browser Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdShowHiddenMembersC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdShowHiddenMembersS"}

Shows and hides the hidden members in the Object Browser for each class. Hidden members are members that are not intended for use by the programmer of the reusable object and are not normally visible in the Object Browser. They appear as light gray text.

Customize Command (Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdCustomizeContextC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdCustomizeContextS"}

Displays the Customize dialog box where you can add, delete, modify, or create custom toolbars.

Delete Watch Command (Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdDeleteWatchContextC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdDeleteWatchContextS"}

Deletes the selected watch expression.

Find Whole Word Only Command (Object Browser Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdFindWholeWordOnlyContextC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdFindWholeWordOnlyContextS"}

Allows you to search for occurrences that exactly match the word you typed into the Search Text box of the Object Browser.

Hide Command (Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdHideContextC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdHideContextS"}

Hides the active window, project, folder, module, or the Toolbox.

Collapse Parent Command (Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdCollapseParentContextC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdCollapseParentContextS"}

Shrinks the list of items in the Locals or Watch window to the parent item. When you collapse a list, the icon to the left of the expression changes from a minus sign (-) to a plus sign (+).

Properties Command (Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdPropertiesContextC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdPropertiesContextS"}

Opens the Properties window for the selected item.

Toggle Commands (Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdToggleContextC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdToggleContextS"}

Breakpoint Sets or removes a breakpoint at the current line.

Bookmark Displays or removes a bookmark at the active line in the Code window.

Move Command (Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdMoveShortcutC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdMoveShortcutS"}

Depending on whether a window or the Toolbox is active when you select the Move command, one of the following actions occurs:

- Changes the pointer to a  so you can move the active window to another location.
- Displays the Page Order dialog box where you can change the order of the pages in your Toolbox.

  You can move the selected page to the left or right of the other pages.

Comment Block and Uncomment Block Commands

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdCommentBlockC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdCommentBlockS"}

Adds and removes the comment character, an apostrophe, for each line of a selected block of text. If you do not have text selected and you choose the Comment Block or Uncomment Block command, the comment character is added or removed in the line where the pointer is located.

Comment Block Adds the comment character to each line of a selected block of text.

Uncomment Block Removes the comment character from each line of a selected block of text.

Properties Command (Object Browser Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdPropertiesCommandObjectBrowserShortcutMenuC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdPropertiesCommandObjectBrowserShortcutMenuS"}

Displays the Member Options dialog box where you type a description of the custom member and give it a Help Context ID and Help File name.

Only available for items containing Basic code.

Help Command (Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdHelpShortcutC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdHelpShortcutS"}

Displays the help topic related to the item selected when the Help command is chosen.

View Object and View Code Commands (Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdViewObjectShortcutC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdViewObjectShortcutS"}

Allows you to view either the Design window or the Code window.

View Object Displays or activates the selected item.

View Code Displays or activates the Code window for a currently selected object.

Dockable Command (Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdDockingViewCommandShortcutC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdDockingViewCommandShortcutS"}

Toggles the ability to dock the active window on and off.

A window is docked when it is attached or "anchored" to one edge of the screen, application window, or another dockable window. When you move a dockable window, it "snaps" to the location. A window is not dockable when you can move it anywhere on the screen and leave it there.

Group Members Command (Object Browser Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdGroupMembersShortcutC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdGroupMembersShortcutS"}

Toggles the Members of pane between an alphabetical list of the members of the selected class and a list grouped by the member type.

View Definition Command (Object Browser Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdViewDefintionShortcutC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdViewDefintionShortcutS"}

Takes you to the definition of the selected member.

Size Command (Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdSizeShortcutC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdSizeShortcutS"}

Changes the pointer so that you can resize the active window or Toolbox.

Copy Command (Object Browser Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdCopyOBShortcutC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdCopyOBShortcutS"}

Copies the text for the selected class or member to the Clipboard.

Wildcard Characters used in String Comparisons

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

Built-in pattern matching provides a versatile tool for making string comparisons. The following table shows the wildcard characters you can use with the Like operator and the number of digits or strings they match.

Character(s) in *pattern*	**Matches in *expression***
?	Any single character
*	Zero or more characters
#	Any single digit (09)
[*charlist*]	Any single character in *charlist*
[!*charlist*]	Any single character not in *charlist*

A group of one or more characters (*charlist*) enclosed in brackets ([ ]) can be used to match any single character in *expression* and can include almost any characters in the ANSI character set, including digits. In fact, the special characters opening bracket ([ ), question mark (?), number sign (#), and asterisk (*) can be used to match themselves directly only if enclosed in brackets. The closing bracket ( ]) can't be used within a group to match itself, but it can be used outside a group as an individual character.

In addition to a simple list of characters enclosed in brackets, *charlist* can specify a range of characters by using a hyphen (-) to separate the upper and lower bounds of the range. For example, using [A-Z] in *pattern* results in a match if the corresponding character position in *expression* contains any of the uppercase letters in the range A through Z. Multiple ranges can be included within the brackets without any delimiting. For example, [a-zA-Z0-9] matches any alphanumeric character.

Other important rules for pattern matching include the following:

- An exclamation mark (!) at the beginning of *charlist* means that a match is made if any character except those in *charlist* are found in *expression*. When used outside brackets, the exclamation mark matches itself.
- The hyphen (-) can be used either at the beginning (after an exclamation mark if one is used) or at the end of *charlist* to match itself. In any other location, the hyphen is used to identify a range of ANSI characters.
- When a range of characters is specified, they must appear in ascending sort order (A-Z or 0-100). [A-Z] is a valid pattern, but [Z-A] isn't.
- The character sequence [ ] is ignored; it's considered to be a zero-length string ("").

Close Command (Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdCloseContextC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdCloseContextS"}

Closes the selected window.

Step Into, Step Over, and Step Out Commands (Debug Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdStepInC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdStepInS"}

Step Into Executes code one statement at a time.

Step Into executes the statement at the current execution point. If the statement is a call to a procedure, the next statement displayed is the first statement in the procedure.

At design time, this menu item begins execution and enters break mode before the first line of code is executed. Not available at run time.

Toolbar button:  . Keyboard shortcut: F8.

Step Over Similar to Step Into. The difference in use occurs when the current statement contains a call to a procedure.

Step Over executes the procedure as a unit, and then steps to the next statement in the current procedure. Therefore, the next statement displayed is the next statement in the current procedure regardless of whether the current statement is a call to another procedure. Available in break mode only.

Toolbar button:  . Keyboard shortcut: SHIFT+F8.

Step Out Executes the remaining lines of a function in which the current execution point lies. The next statement displayed is the statement following the procedure call. All of the code is executed between the current and the final execution points. Available in break mode only.

Toolbar button:  . Keyboard shortcut: CTRL+SHIFT+F8.

Run to Cursor Command (Debug Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdRunToCursorC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdRunToCursorS"}

When your application is in break mode, use Run To Cursor to select a statement further down in your code where you want execution to stop. Your application will run from the current statement to the selected statement and the current line of execution margin indicator,  , appears in the Margin Indicator bar.

You can use this command, for example, to avoid stepping through large loops.

Available only in break mode.

Keyboard shortcut: CTRL+F8.

Toggle Breakpoint Command (Debug Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdToggleBreakpointC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdToggleBreakpointS"}

Sets or removes a breakpoint at the current line. You can't set a breakpoint on lines containing nonexecutable code such as comments, declaration statements, or blank lines.

A line of code in which a breakpoint is set appears in the colors specified in the Editor Format tab of the Options dialog box.

Not available at run time.

Toolbar shortcut:  . Keyboard shortcut: F9.

Clear All Breakpoints Command (Debug Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdClearAllBreakpointsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdClearAllBreakpointsS"}

Removes all breakpoints in your project. Your application may still interrupt execution, however, if you have set a watch expression or selected the Break on All Errors option in the General tab of the Options dialog box. You cannot undo the Clear All Breakpoints command. Not available at run time.

Toolbar shortcut:  . Keyboard shortcut: CTRL+SHIFT+F9.

Set Next Statement Command (Debug Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdSetNextStatementC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdSetNextStatementS"}

Sets the execution point to the line of code you choose. You can set a different line of code to execute after the currently selected statement by selecting the line of code you want to execute and choosing the Set Next Statement command or by dragging the Current Execution Line margin indicator to the line of code you want to execute.

Using Set Next Statement, you can choose a line of code located before or after the currently selected statement. When you run the code, any intervening code isn't executed. Use this command when you want to rerun a statement within the current procedure or to skip over statements you don't want to execute. You can't use Set Next Statement for statements in different procedures.

Toolbar shortcut:  . Keyboard shortcut: CTRL+F9.

Show Next Statement Command (Debug Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdShowNextStatementC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdShowNextStatementS"}

Highlights the next statement to be executed. Use the Show Next Statement command to place the cursor on the line that will execute next.

Available only in break mode.

Toolbar shortcut:  .

Add Watch Command (Debug Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdAddWatchC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdAddWatchS"}

At design time or in break mode, this command displays the Add Watch dialog box in which you enter a watch expression. The expression can be any valid Basic expression. Watch expressions are updated in the Watch window each time you enter break mode.

Toolbar shortcut:  .

Edit Watch Command (Debug Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdEditWatchC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdEditWatchS"}

Displays the Edit Watch dialog box in which you can edit or delete a watch expression. Available when the watch is set even if the Watch window is hidden. Not available at run time.

Toolbar shortcut:  . Keyboard shortcut: CTRL+W.

Quick Watch Command (Debug Menu)

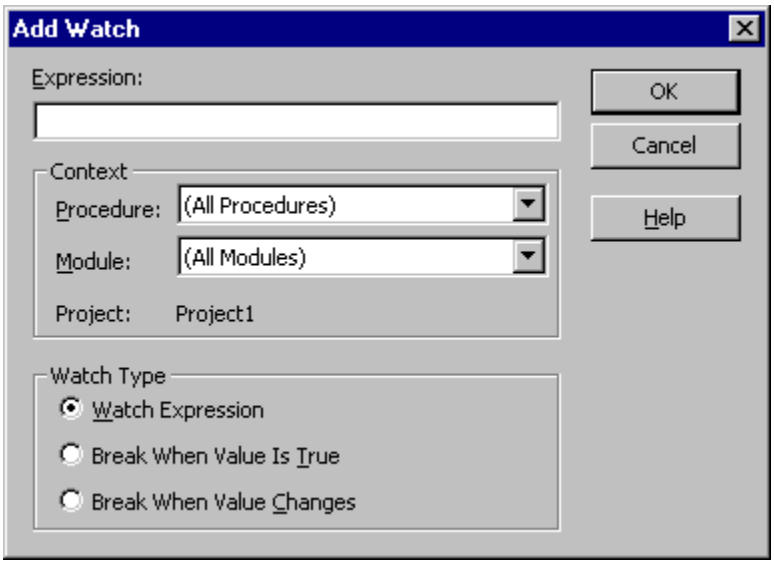
{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdInstantWatchC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdInstantWatchS"}

Displays the Quick Watch dialog box with the current value of the selected expression. Available only in break mode. Use this command to check the current value of a variable, property, or other expression for which you have not defined a watch expression. Select the expression from either the Code window or the Immediate window, and then choose the Quick Watch command. To add a watch expression based on the expression in the Quick Watch dialog box, choose the Add button.

Toolbar shortcut:  . Keyboard shortcut: SHIFT+F9.

Add Watch Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnAddWatchC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnAddWatchS"}



Use to enter a watch expression. The expression can be a variable, a property, a function call, or any other valid Basic expression. Watch expressions are updated in the Watch window each time you enter break mode or after execution of each statement in the Immediate window.

You can drag selected expressions from the Code window into the Watch window.

Important When selecting a context for a watch expression, use the narrowest scope that fits your needs. Selecting all procedures or all modules could slow down execution considerably, since the expression is evaluated after execution of each statement. Selecting a specific procedure for a context affects execution only while the procedure is in the list of active procedure calls, which you can see by choosing the Call Stack command on the View menu.

Dialog Box Options

Expression Displays the selected expression by default. The expression is a variable, a property, a function call, or any other valid expression. You may enter a different expression to evaluate.

Context Sets the scope of the variables watched in the expression.

- Procedure — Displays the procedure name where the selected term resides (default). Defines the procedure(s) in which the expression is evaluated. You may select all procedures or a specific procedure context in which to evaluate the variable.
- Module — Displays the module name where the selected term resides (default). You may select all modules or a specific module context in which to evaluate the variable.
- Project — Displays the name of the current project. Expressions can't be evaluated in a context outside of the current project.

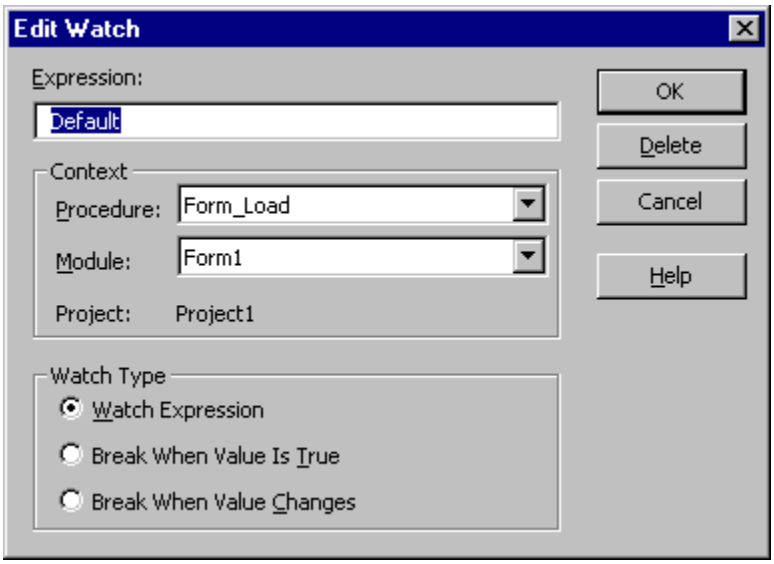
Watch Type Determines how Visual Basic responds to the watch expression.

- Watch Expression — Displays the watch expression and its value in the Watch window. When you enter break mode, the value of the watch expression is automatically updated.
- Break When Value Is True — Execution automatically enters break mode when the expression evaluates to true or is any nonzero value (not valid for string expressions).
- Break When Value Changes — Execution automatically enters break mode when the value of

expression changes within the specified context.

Edit Watch Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnEditWatchC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnEditWatchS"}



Use to delete or edit the context or type of a watch expression.

Important When selecting a context for a watch expression, use the narrowest scope that fits your needs. Selecting all procedures or all modules could slow down execution considerably, since the expression is evaluated after execution of each statement. Selecting a specific procedure for a context affects execution only while the procedure is in the list of active procedure calls.

Dialog Box Options

Expression Displays the selected expression by default. The expression is a variable, a property, a function call, or any other valid expression. You may enter a different expression to evaluate.

Context Sets the scope of the variables watched in the expression.

- Procedure — Displays the procedure name where the selected term resides (default). Defines the procedure(s) in which the expression is evaluated. You may select all procedures or a specific procedure context in which to evaluate the variable.
- Module — Displays the module name where the selected term resides (default). You may select all modules or a specific module context in which to evaluate the variable.
- Project — Displays the name of the current project. Expressions can't be evaluated in a context outside of the current project.

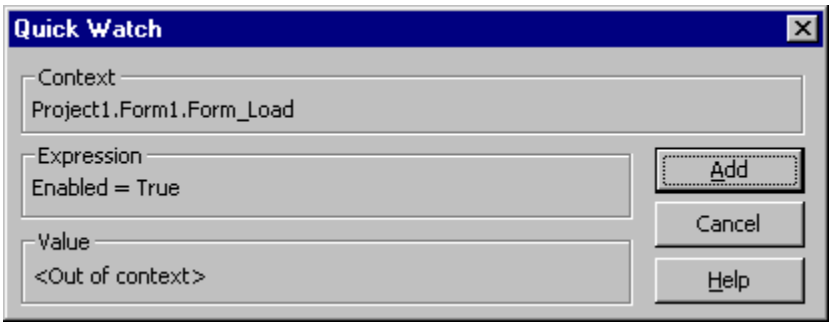
Watch Type Determines how Visual Basic responds to the watch expression.

- Watch Expression — Displays the watch expression and its value in the Watch window. When you enter break mode, the value of the watch expression is automatically updated.
- Break When Value Is True — Execution automatically enters break mode when the expression evaluates to true or is any nonzero value (not valid for string expressions).
- Break When Value Changes — Execution automatically enters break mode when the value of expression changes within the specified context.

Delete Deletes the watch expression.

Quick Watch Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnInstantWatchC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnInstantWatchDialogBoxS"}



Displays the current value of a selected expression. This functionality is useful when debugging your code if you want to see the current value of a variable, property, or other expression.

Dialog Box Options

Current Context Lists the names of the project, module, and procedure where the watch expression resides.

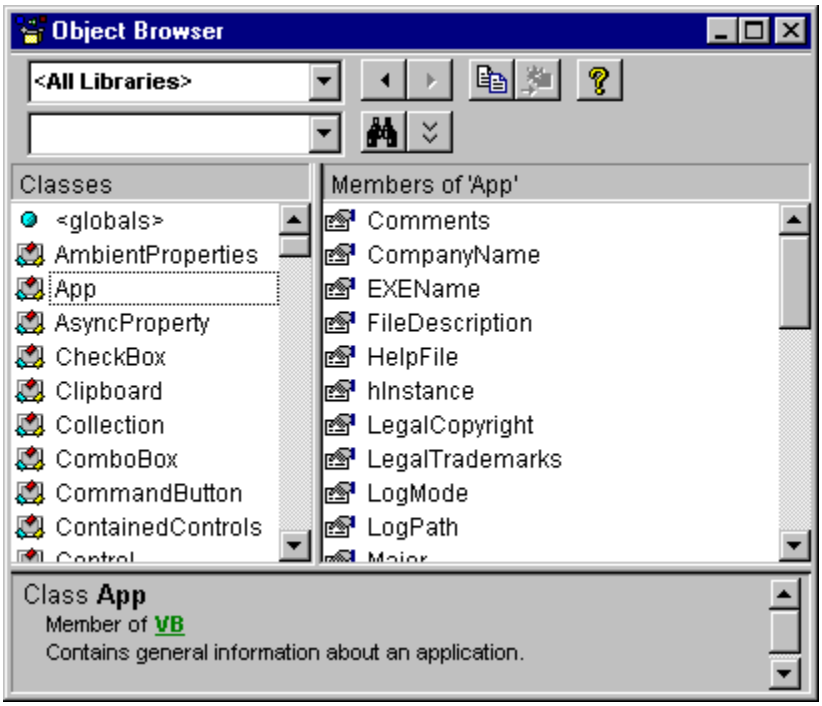
Expression Shows the selected expression.

Value Shows the value of the selected expression. The current value isn't displayed if the expression context isn't within a procedure listed in the Calls dialog box.

Add Adds the expression to the Watch window.

Object Browser

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnObjectBrowserC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnObjectBrowserS"}



Displays the classes, properties, methods, events, and constants available from object libraries and the procedures in your project. You can use it to find and use objects you create, as well as objects from other applications.

You can get help for the Object Browser by searching for Object Browser in Help.

Window Elements

Project/Library Box

Displays the currently referenced libraries for the active project. You can add libraries in the References dialog box. <All Libraries> allows all of the libraries to be displayed at one time.

Search Text Box

Contains the string that you want to use in your search. You can type or choose the string you want. The Search Text box contains the last 4 search strings that you entered until you close the project. You can use the standard Visual Basic wildcards when typing a string.

If you want to search for a whole word, you can use the Find Whole Word Only command from the shortcut menu.

Go Back Button

Allows you to go back to the previous selection in the Classes and Members of lists. Each time you click it you move back one selection until all of your choices are exhausted.

Go Forward Button

Allows you to repeat your original selections in the Classes and Members of lists each time you click it, until you exhaust the list of selections.

Copy to Clipboard Button

Copies the current selection in the Members of list or the Details pane text to the clipboard. You can then paste the selection into your code.

Show Definition Button

Moves the cursor to the place in the Code window where the selection in the Members of list or the Classes list is defined.

Help Button

Displays the online Help topic for the item selected in the Classes or the Members of list. You can also use F1.

Search Button

Initiates a search of the libraries for the class or property, method, event or constant that matches the string you typed in the Search Text box, and opens the Search Results pane with the appropriate list of information.

Show/Hide Search Results Button

Opens or hides the Search Results pane. The Search Results pane changes to show the search results from the project or library chosen in the Project/Library list. Search results are listed alphabetically from A to Z.

Search Results List

Displays the library, class, and member that corresponds to the items that contain your search string. The Search Results pane changes when you change the selection in the Project/Library box.

Classes List

Displays all of the available classes in the library or project selected in the Project/Libraries box. If there is code written for a class, that class appears in bold. The list always begins with <globals>, a list of globally accessible members.

If you select a Class and do not specify a member, you will get the default member if one is available. The default member is identified by an asterisk (*) or by the default icon specific to the member.

Members of List

Displays the elements of the class selected in the Classes pane by group and then alphabetically within each group. Methods, properties, events, or constants that have code written for them appear bold. You can change the order of this list with the Group Members command on the Object Browser shortcut menu.

Details Pane

Shows the definition of the member. The Details pane contains a jump to the class or library to which the element belongs. Some members have jumps to their parent class. For example, if the text in the Details pane states that Command1 is declared as a command button type, clicking on command button takes you to the Command Button class.

You can copy or drag text from the Details pane to the Code window.

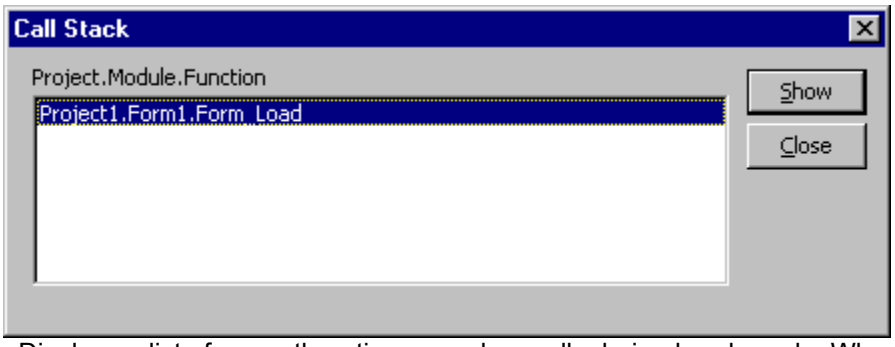
Split Bar

Splits the panes so that you can adjust their size. There are splits between the:

- Classes box and the Members of box.
- Search Results list and the Classes and Members of boxes.
- Classes and Members of boxes and the Details pane.

Calls Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnCallsDialogC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnCallsDialogS"}



Displays a list of currently active procedure calls during break mode. When executing code in a procedure, that procedure is added to a list of active procedure calls. Each time a procedure calls another procedure, it is added to the list. Called procedures are removed from the list when execution returns to the calling procedure. Procedures called from the Immediate window are also added to the calls list.

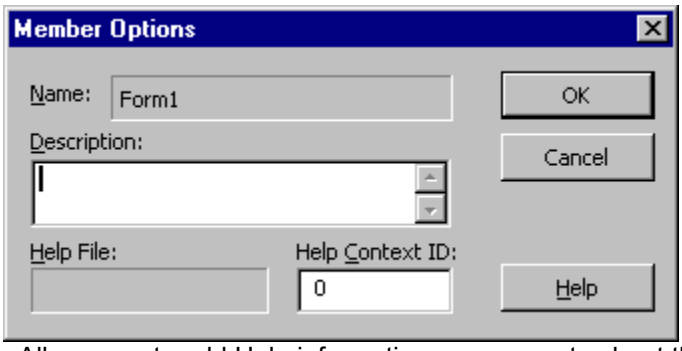
Dialog Box Options

Project Module Function Lists the procedures.

Show Moves the insertion point to the location where the call was made and turns on the Call Stack indicator, ➪.

Member Options Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnMemberOptionsC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnMemberOptionsS"}



Allows you to add Help information or comments about the procedures, that you define in your project.

Dialog Box Options

Member Name Displays the name of the item selected in the Classes or the Members of list.

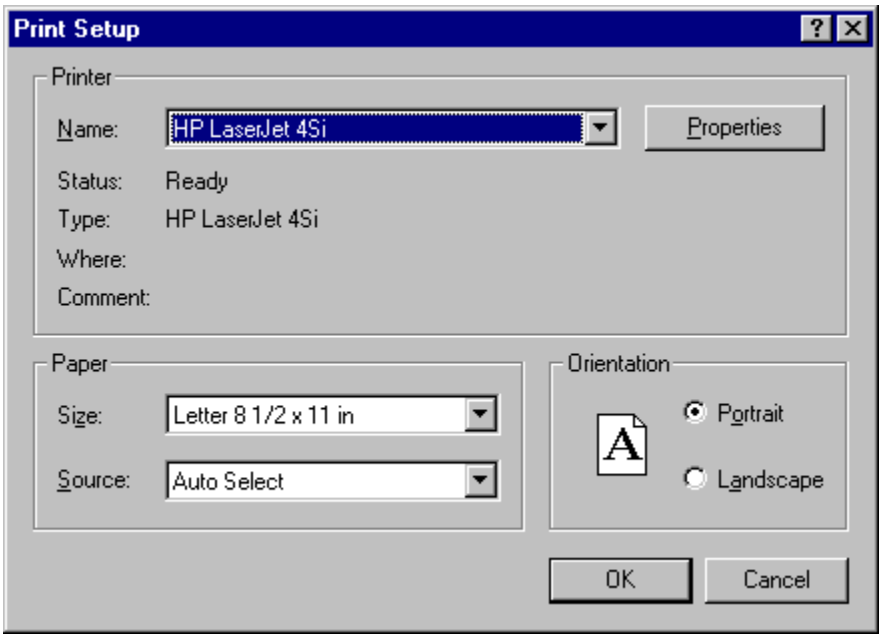
Description Allows you to specify a string that shows up in the Details pane of the Object Browser.

Help File Displays the Help file associated with the item listed in the Member Name box. This is set in the Help File Name box in the **General** tab of the Project Properties dialog box.

Help Context ID Assigns a unique numeric value for the context ID. This value is used to find the appropriate Help topic in the file listed in the Help File box when the user presses F1 or the  button while looking at the procedure in the Object Browser.

Print Setup Dialog Box

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



Appears whenever you select the Printer Setup command.

Use the Print Setup dialog box to select the printer, page orientation, and paper size.

Dialog box options

Printer Allows you to specify the printer. If you don't select a printer, Visual Basic will print to the Windows default printer.

- Name — Displays a list of available printers.
- Status — Displays the status of the printer and whether it is ready to print.
- Type — Displays the type of printer.
- Where — Displays the location of the printer. If the printer is on a network, displays the path to the server.
- Comment — Displays the physical location of the printer and additional information.
- Properties — Opens the Properties dialog box specific to the printer where you can choose additional options such as paper and the way graphics are printed.

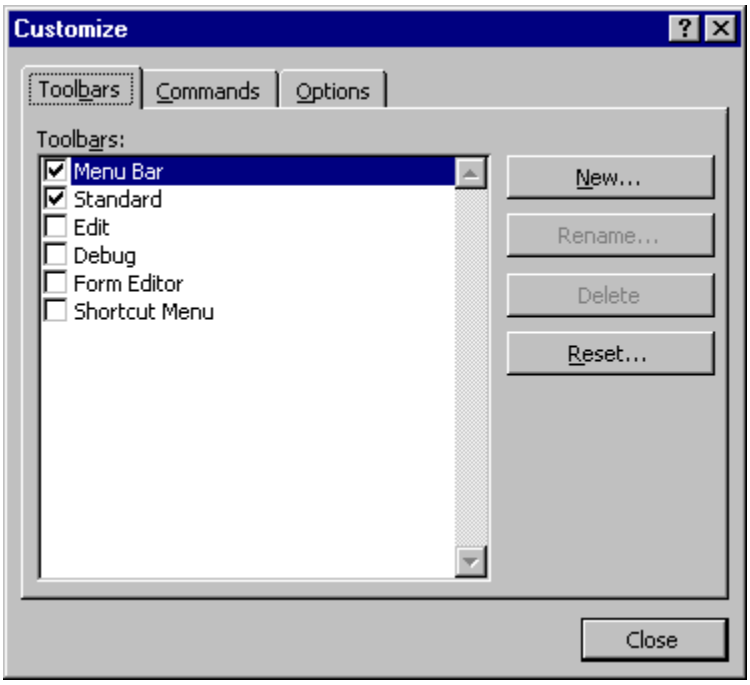
Paper Allows you to select the paper size and source (from among those available for the printer). The sizes and sources available depend on the printer you have selected and they change when you change printers.

- Size — Displays a list of the available paper sizes.
- Source — Displays the available source of paper for the printer you choose.

Orientation Allows you to specify whether the program is to print in Portrait or Landscape orientation.

Customize Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vadlgCustomizeDialogBoxC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vadlgCustomizeDialogBoxS"}



Allows you to customize your toolbars and menus.

Tabs

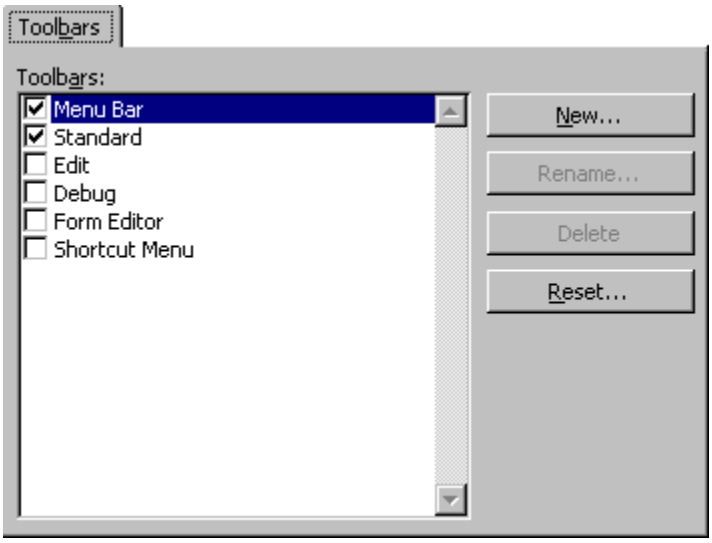
Toolbars Allows you to create, rename, delete, and reset your toolbars.

Commands Contains commands you can drag to your menus and toolbars.

Options Allows you to change the size of your toolbar buttons, to show ToolTips and shortcut keys, and to add animation to your menus.

Toolbars Tab (Customize Dialog Box)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vadlgToolbarsTabCustomizeDialogC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vadlgToolbarsTabCustomizeDialogS"}



Allows you to create, rename, delete, and reset your toolbars,

Tab Options

Toolbars box Displays the toolbars built into Visual Basic and any toolbars you create. When you show a toolbar, a check mark appears to the left of it.

Note The Menu bar cannot be hidden. It can only be reset.

New Opens the New Toolbar dialog box where you type the name for your new toolbar in the Toolbar Name box.

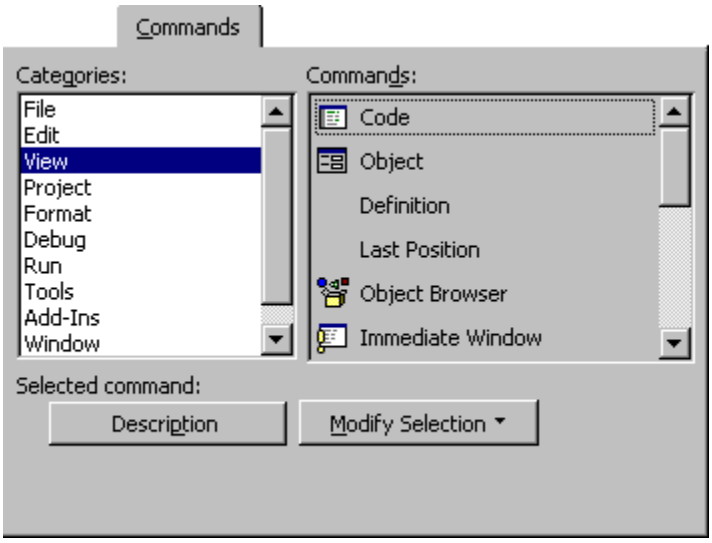
Rename Opens the Rename Toolbar dialog box where you type the new name for your toolbar. Only available if you select a user-defined toolbar.

Delete Deletes a user-defined toolbar from your project. Only available if you select a user-defined toolbar.

Reset Removes any changes to the built-in toolbars and resets them to their original state. Only available if you select a built-in toolbar.

Commands Tab (Customize Dialog Box)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vadlgCommandsTabCustomizeDialogC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vadlgCommandsTabCustomizeDialogS"}



Allows you to add controls and modify existing controls on the Menu bar or any toolbar. You can also modify the button image and text of your added commands.

Tab Options

Categories Lists the different command categories.

Commands Lists the controls available for the category you select in the Categories list. You can drag the command to the toolbar where you want the command to reside. To add the command to a menu, drag it over the menu's title and then into the location in the menu that appears.

Description Displays a QuickTip for the currently selected control.

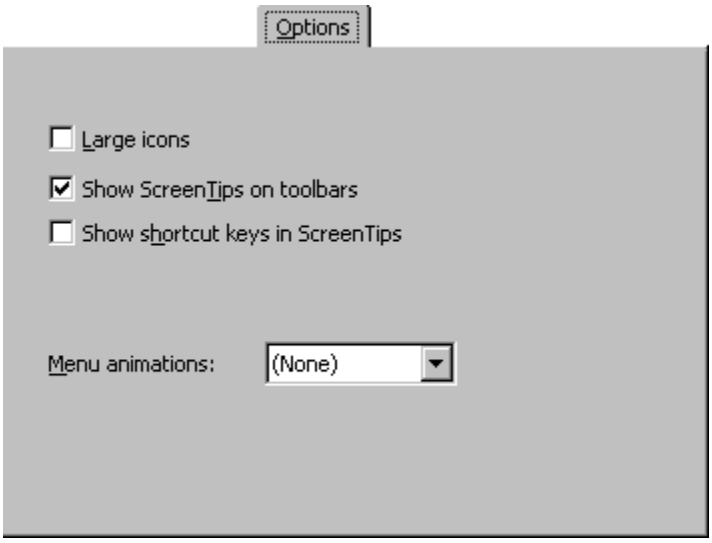
Modify Selection Allows you to change the selected command.

- Reset — Resets the command to the default.
- Delete — Deletes the command.
- Name — Changes the name of the control to the name you type into the box. By using the ampersand (&), you can also set shortcut keys.
- Copy Button Image — Copies the button image to the Clipboard.
- Paste Button Image — Pastes the button image from the Clipboard.
- Reset Button Image — Resets the button image to its default setting.
- Edit Button Image — Displays the Button Editor dialog box.
- Change Button Image — Displays a list of button images from which you can select a different image for your button.
- Default Style — For toolbar buttons, shows only the icon. For menu items, shows the icon and the name.
- Text Only (Always) — Shows the command name only.
- Text Only (in Menus) — Hides the icon, if any, for menu items. This options has no effect on toolbar buttons.
- Image and Text — For toolbars, shows both the icon and name. This option has no effect on menu items.

- Begin a Group — For toolbars, puts a separator line before the control. Dimmed when the control is at the beginning of the toolbar.

Options Tab (Customize Dialog Box)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vadlgOptionsTabCustomizeDialogBoxC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vadlgOptionsTabCustomizeDialogBoxS"}



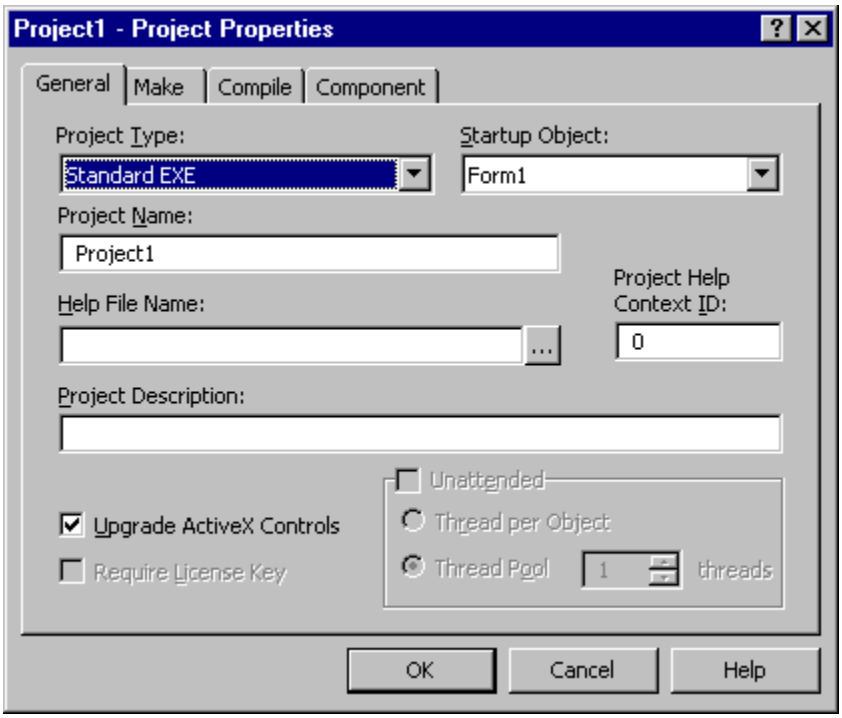
Allows you to change the appearance of your menu bar and toolbars.

Tab Options

Large Icons Changes the toolbar icons to a larger size.
Show ScreenTips on toolbars Turns ScreenTips off and on.
Show shortcut keys in ScreenTips Displays the shortcut keys on the menus.
Menu animations Lists available animations you can apply to your menu bar.

Project Properties Dialog Box

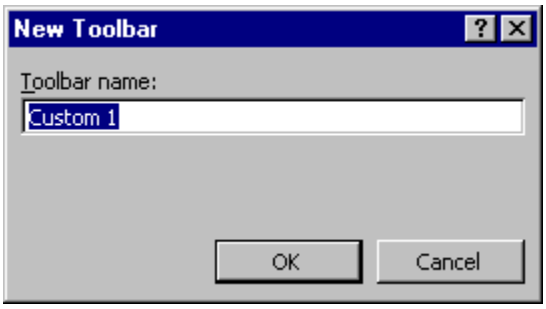
{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgProjectPropertiesDialogC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgProjectPropertiesDialogS"}



Specifies the settings for a specific project.

New Toolbar Dialog Box

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



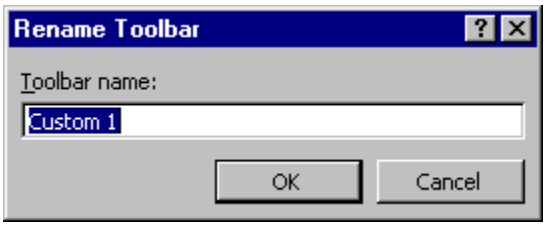
Allows you to give your new toolbar a name.

Dialog Box Option

Toolbar name You can type the name for your new toolbar or use the default name. The default name changes for each custom toolbar, for example, Custom1, Custom2 and so on.

Rename Toolbar Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgRenameToolbarShortcutC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgRenameToolbarShortcutS"}



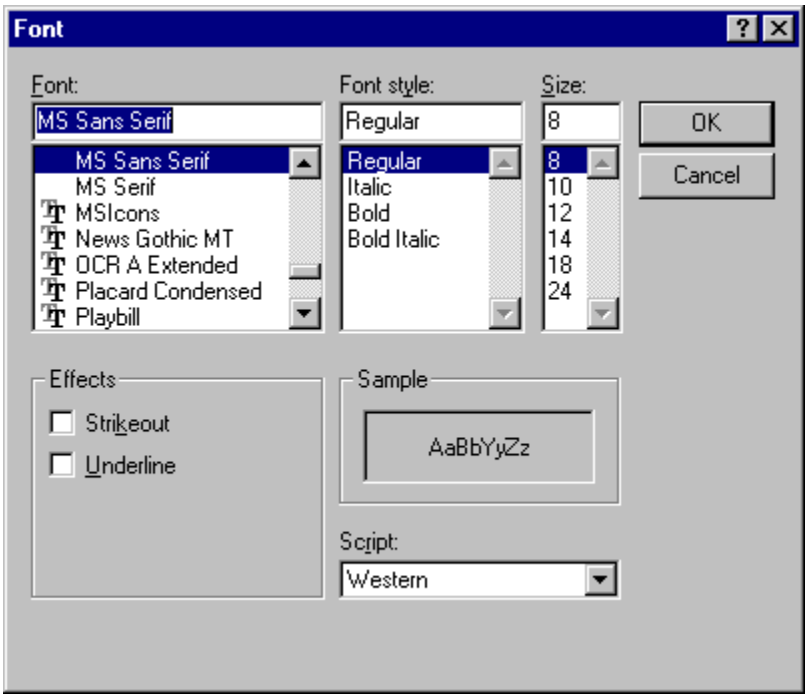
Allows you to rename your custom toolbars.

Dialog Box Option

Toolbar name Type a new name for your toolbar.

Font Dialog Box

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



Use to change the fonts, the font size, and/or the font style you use for text and data fields.

Dialog Box Options

Font Lists all the fonts that you have installed for Windows. When the box first appears, the font already in use for the selected element is highlighted.

Font Style Lists four additional attributes that you can assign to the font selected:

- Regular — Standard, unmodified style
- Bold — Boldface
- Italic — Italic
- Bold Italic — Bold Italic

Size Lists common point sizes for the highlighted font. When the box first appears, the point size for the font already in use for the selected element is highlighted, and the highlighted point size appears in the edit box at the top.

You can select directly from the list or type the new point size in the edit box at the top (if you know that you have additional sizes installed for the currently selected printer or if you are using scalable type).

Effects Lists two additional options that you can use for highlighting the selected font.

- Strikeout — Prints the strikeout character across the font
- Underline — Underlines the font

You may select as many of the Effects as you wish.

Sample Displays a sample of the font you have selected. The sample shows the font, style, size, effects, and color you have specified. You can use this box to preview the results as you experiment with different formatting options.

Script Displays a list of available scripts.

OK Applies the font changes to the selected report element.

Cancel Cancels all font changes and leaves the report unchanged.

Icons Used in the Object Browser and Code Windows

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

The Object Browser and Code window contain many icons that represent classes and members. The following is a list of icons and what they represent.

This Icon:	Represents a:
	Property
	Default Property
	Method
	Default Method
	Event
	Constant
	Module
	Class
	User Defined Type
	Global
	Library
	Project
	Built-in keywords and types
	Enum

Redo and Undo Commands (Edit Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdCut;vbcmdUndoC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdUndoS"}

Undo Reverses the last editing action, such as typing text in the code window or deleting controls. When you delete one or more controls, you can use the Undo command to restore the controls and all their properties. For forms, Undo is unavailable if the form has been edited since the last control deletion.

Toolbar shortcut  . Keyboard shortcuts: CTRL+Z or ALT+BACKSPACE.

Note You can't undo a Cut operation using the Undo command.

Redo Restores the last text editing if no other actions have occurred since the last Undo.

Toolbar shortcut  .

For text edits, you can use Undo and Redo to restore up to twenty edits.
These commands are unavailable at runtime, or if there was no previous edit, or if any other action has been performed after the last edit. Also, some large edits may cause low memory conditions that could prevent an Undo action.

Cut, Copy, Paste, and Delete Commands (Edit Menu)

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

Cut Removes the selected control or text and places it on the Clipboard. You must select at least one character or control for this command to be available. You can undo the Cut command only in the Code window.

Toolbar shortcut:  . Keyboard shortcut: CTRL+X or SHIFT+DELETE

Copy Copies the selected control or text onto the Clipboard. You must select at least one character or control for this command to be available. You cannot undo the Copy command in the Code window.

Toolbar shortcut:  . Keyboard shortcut: CTRL+C or CTRL+INSERT.

Paste Inserts the contents of the Clipboard at the current location. Text is placed at the insertion point.

Pasted controls are placed at the upper-left corner of the active form. You can keep the same control name and create a control array, or paste the control with a different name.

You can undo the Paste command only in the Code window.

Toolbar shortcut:  . Keyboard shortcut: CTRL+V or SHIFT+INS.

Delete Deletes the currently selected control, text, or watch expression. You can undo the Delete command only in the Code window.

Note To delete a file from your disk, use the standard deletion procedures for your operating system.

Toolbar shortcut:  . Keyboard shortcut: DEL.

Not available at run time.

Find and Find Next Commands (Edit Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdFindC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdFindS"}

Find Searches for the specified text in a search range specified in the Find dialog box.

If a search is successful, the Find dialog box closes and Visual Basic selects the located text. If no match is found, Visual Basic displays a message stating that the text was not found.

Toolbar shortcut:  . Keyboard shortcut: CTRL+F.

Find Next Finds and selects the next occurrence of the text specified in the Find What box of the Find dialog box.

Toolbar shortcut:  . Keyboard shortcuts: F3 (Find Next) or SHIFT+F3 (Find Previous).

Dialog Box Options

Find What Type or insert the information you want to find, or click the down arrow and select from the last four entries. If any text is selected or the cursor is on a word when you choose the command, this text is displayed in the Find What box.

Search Specifies the search range.

- Current Procedure—Searches only the current procedure.
- Current Module—Searches only the current module.
- Current Project—Searches all the modules in your project.
- Selected Text—Searches a selected range of code in your project.

Direction Sets the direction of the search to Down or Up from the location of the cursor, or All in the selected search range.

Find Whole Word Only Searches for the full word by itself and not as part of a larger word.

Match Case Finds all occurrences with the exact combination of uppercase and lowercase letters specified in the Find What box.

Use Pattern Matching Searches using pattern-matching characters.

Find Next Finds and selects the next occurrence of the text specified in the Find What box.

Cancel Closes the dialog box without performing the search.

Replace Displays the Replace dialog box, retaining the information typed in the Find What dialog box.

Replace Command (Edit Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdReplaceC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdReplaceS"}

Searches code in the project for the specified text and replaces it with the new text specified in the Replace dialog box.

Toolbar shortcut:  . Keyboard shortcut: CTRL+H.

Dialog Box Options

Find What Type or insert the information you want to find, or click the down arrow and select from the last four entries. If any text is selected or the cursor is on a word when you choose the command, this text is displayed in the Find What box.

Replace With Type the text you want to use as the replacement text or paste it from the Clipboard. To delete the text in the Find What box from the document, leave the Replace With box empty.

Search Specifies the search range.

- Current Procedure—Searches only the current procedure.
- Current Module—Searches only the current module.
- Current Project—Searches all the modules in your project.
- Selected Text—Searches a selected range of code in your project.

Direction Sets the direction of the search to Down, Up, or All in the selected search range.

Find Whole Word Only Searches for the full word by itself and not as part of a larger word.

Match Case Finds all occurrences with the exact combination of uppercase and lowercase letters specified in the Find What box.

Use Pattern Matching Searches using pattern-matching characters.

Find Next Finds and selects the next occurrence of the text specified in the Find What box.

Cancel Closes the dialog box without replacing text.

Replace Confirms before replacing the search text with the replacement text.

Replace All Replaces all occurrences of the search text with the replacement text without stopping for confirmation.

Indent Command (Edit Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdIndentC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdIndentS"}

Shifts all lines in the selection to the next tab stop. All lines in the selection are moved the same number of spaces to retain the same relative indentation within the selected block.

You can change the tab width on the Editor tab of the Options dialog box.

Toolbar shortcut:  . Keyboard shortcuts: CTRL+M.

Outdent Command (Edit Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdOutdentC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdOutdentS"}

Shifts all lines in the selection to the previous tab stop. All lines in the selection are moved the same number of spaces to retain the same relative indentation within the selected block.

You can change the tab width on the Editor tab of the Options dialog box.

Toolbar shortcut:  . Keyboard shortcuts: CTRL+SHIFT+M.

Bookmarks Commands (Edit Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdBookmarksC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdBookmarksS"}

Displays a menu that you can use to create or remove placeholders in the Code window, move to the next or preceding bookmark, or clear all of the bookmarks.

When you add a bookmark, a  appears next to the line where the bookmark is inserted.

Toggle Bookmark Toggles a bookmark on or off.

Toolbar button: 

Next Bookmark Moves the insertion point to the next bookmark.

Toolbar button: 

Previous Bookmark Moves the insertion point to the previous bookmark.

Toolbar button: 

Clear All Bookmarks Removes all bookmarks.

Toolbar button: 

Select All Command (Edit Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdSelectAllC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdSelectAllS"}

Selects all of the code in the active Code window.

List Properties/Methods Command (Edit Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdSelectMemberC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdSelectMemberS"}

Opens a dropdown list box in the Code window that contains the properties and methods available for the object that precedes the period (.). The List Properties/Methods command also displays a list of the globally available methods when the pointer is on a blank space. To have the list box automatically open as you type your code, select Auto List Members on the Editor tab in the Options dialog box.

You can find the property or method you want in the list box by:

- Typing the name. As you type, the property or method that matches the characters you type is selected and moves to the top of the list.
- Using the up and down arrow keys to move up and down in the list.
- Scrolling through the list and selecting the property or method you want.

You can insert the property or method into your statement by:

- Double-clicking the property or method.
- Selecting the property or method and pressing TAB to insert the selection or pressing ENTER to insert the selection and move to the next line.

Note Objects of the type Variant do not show a list after the period (.).

Toolbar shortcut:  . Keyboard shortcut: CTRL+J.

List Constants Command (Edit Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdSelectConstantC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdSelectConstantS"}

Opens a dropdown list box in the Code window that contains the valid constants for a property that you typed, and that preceded the equal sign (=). The List Constants command also works for functions with arguments that are constants. To have the list box automatically open as you type your code, select Auto List Members on the Editor tab in the Options dialog box.

You can find the constant you want by:

- Typing the name.
- Using the up and down arrow keys to move up and down in the list.
- Scrolling through the list and selecting the constant you want.

You can insert the constant into your code statement by:

- Double-clicking the constant.
- Selecting the constant and pressing TAB to insert the selection or pressing ENTER to insert the selection and move to the next line.

Toolbar shortcut:  . Keyboard shortcut: CTRL+SHIFT+J.

Quick Info Command (Edit Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdQuickInfoC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdQuickInfoS"}

Provides the syntax for a variable, function, statement, method, or procedure selected in the Code window.

Quick Info shows the syntax for the item and highlights the current parameter. For functions and procedures with parameters, the parameter appears bold as you type it, until you type the comma used to delineate it from the next parameter.

To have Quick Info automatically appear as you type your code, select Auto Quick Info on the Editor tab in the Options dialog box.

Toolbar shortcut:  . Keyboard shortcut: CTRL+I.

Parameter Info Command (Edit Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdParameterInfoC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdParameterInfoS"}

Shows a popup in the Code window that contains information about the parameters of the initial function or statement. If you have a function or statement that contains functions as its parameters, choosing Parameter Info provides information about the first function. Quick Info provides information about each embedded function.

As you type a parameter it is bold until you type the comma used to delineate it from the next parameter

The Parameter Info, once activated, will not close until:

- All of the required parameters are entered.
- The function is ended without using all of the optional parameters.
- You press ESC.

Toolbar shortcut:  . Keyboard shortcut: CTRL+SHIFT+I.

Complete Word Command (Edit Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdCompleteWordC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdCompleteWordS"}

Fills in the rest of the word you are typing once you have entered enough characters for Visual Basic to identify the word you want.

Toolbar shortcut:  . Keyboard shortcut: CTRL+SPACEBAR.

Print Command (File Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdPrintC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdPrintS"}

Prints forms and code to the printer specified in the Microsoft Windows Control Panel. Available only at design time.

Toolbar shortcut:  . Keyboard shortcut: CTRL+P.

Dialog Box Options

Printer Identifies the printer to which you are printing.

Range Determines the range you print:

- Selection — Prints the currently selected code.
- Current Module — Prints the forms and/or code for the currently selected module.
- Current Project — Prints the forms and/or code for the entire project.

Print What Determines what you print. You can select as many options as you like, depending on what you selected as the Range.

- Form Image — Prints the form images.
- Code — Prints the code for the selected range.

Print Quality Determines whether you print high, medium, low, or draft output quality.

Print to File If selected, print is sent to the file specified in the Print To File dialog box. This dialog box appears after you choose OK in the Print dialog box.

OK Prints your selection.

Cancel Closes the dialog box without printing.

Setup Displays the standard Print Setup dialog box.

Align Commands (Format Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdAlignC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdAlignS"}

Aligns selected objects with each other using the last selected object, the one with the solid color grab handles, as the reference. The color of the grab handles is based on the color you set from Selected Items on the Appearance tab of the Display Properties dialog box in the Control Panel.

Lefts Aligns the horizontal position of the selected objects, putting the left-most edges in line with the last selected object, the one with the solid color grab handles.

Toolbar shortcut:  .

Centers Aligns the horizontal position of the selected objects, putting the centers in line with the last selected object, the one with the solid color grab handles.

Toolbar shortcut:  .

Rights Aligns the horizontal position of the selected objects, putting the right-most edges in line with the last selected object, the one with the solid color grab handles.

Toolbar shortcut:  .

Tops Aligns the vertical position of the selected objects, putting the tops in line with the last selected object, the one with the solid color handles.

Toolbar shortcut:  .

Middles Aligns the vertical position of selected objects, putting the middles in line with the last selected object, the one with the solid color grab handles.

Toolbar shortcut:  .

Bottoms Aligns the vertical position of the selected objects, putting the bottoms in line with the last selected object, the one with the solid color grab handles.

Toolbar shortcut:  .

To Grid Aligns the top left of the selected objects to the closest grid. The object is not resized.

Toolbar shortcut:  .

Make Same Size Commands (Format Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdMakeSameSizeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdMakeSameSizeS"}

Using the last object selected, the object with the solid color grab handles, makes the selected objects the same size in the dimension you select. The color of the grab handles is based on the color you set from Selected Items on the Appearance tab of the Display Properties dialog box in the Control Panel.

Width Adjusts width.

Toolbar shortcut:  .

Height Adjusts height.

Toolbar shortcut:  .

Both Adjusts both the width and the height.

Toolbar shortcut:  .

Horizontal Spacing Commands (Format Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdHorizontalSpacingC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdHorizontalSpacingS"}

Changes the horizontal space between selected objects.

Make Equal Moves the selected objects so that there is equal space between them using the outermost objects as endpoints. The outermost objects do not move.

Toolbar shortcut:  .

Increase Increases horizontal spacing by one grid unit based on the object with focus. When an object has focus, black handles appear on its borders. You can change the size of your grid units in the General tab of the Options dialog box.

Toolbar shortcut:  .

Decrease Decreases horizontal spacing by one grid unit based on the object with focus. When an object has focus, black handles appear on its borders. You can change the size of your grid units in the General tab of the Options dialog box.

Toolbar shortcut:  .

Remove Removes the horizontal space so that the objects are aligned with their edges touching based on the object with focus. When an object has focus, black handles appear on its borders.

Toolbar shortcut:  .

Note The object with focus does not move but the other objects move around it.

Note If using the Horizontal Spacing command does not produce the results you want, try to manually rearrange some of the objects and repeat the command. Also, try the using the Vertical Spacing command.

Vertical Spacing Commands (Format Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdVerticalSpacingC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdVerticalSpacingS"}

Changes the vertical space between the selected objects, based on the object with focus. When an object has focus, black handles appear on its borders.

Make Equal Moves the selected objects so that there is equal space between them using the top and bottoms objects as the end points. The top and bottom objects do not move.

Toolbar shortcut:  .

Increase Increases the vertical spacing by one grid based on the object with focus. You can change the size of your grid units in the General tab of the Options dialog box.

Toolbar shortcut:  .

Decrease Decreases the vertical spacing by one grid based on the object with focus. You can change the size of your grid units in the General tab of the Options dialog box.

Toolbar shortcut:  .

Remove Removes the vertical spacing so that the object's borders are touching, based on the object with focus

Toolbar shortcut:  .

Note The object with focus does not move but the other objects move around it.

Note If using the Vertical Spacing command does not produce the results you want, try to manually rearrange some of the objects and repeat the command. Also, try the using the Horizontal Spacing command.

Center in Form Commands (Format Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdCenterC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdCenterS"}

Centers selected objects on the central axes of the form.

Horizontally Aligns the middles of the selected objects to a horizontal line in the middle of the form.

Toolbar shortcut:  .

Vertically Aligns the centers of the selected objects to a vertical line in the center of the form.

Toolbar shortcut:  .

Size to Grid Command (Format Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdSizeToGridC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdSizeToGridS"}

Adjusts the height and width of the selected object to fit the nearest gridlines in the form. You can change the size of your grid on the General tab of the Options dialog box.

Not available in break mode.

Toolbar shortcut:  .

Order Command (Format Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdOrderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdOrderS"}

Changes the order of the selected objects on a form.

Bring To Front Moves the selected objects to the front of all other objects on a form.

Toolbar shortcut:  . Keyboard shortcut: CTRL+J.

Send To Back Moves the selected objects behind all other objects on a form.

Toolbar shortcut:  . Keyboard shortcut: CTRL+K.

Note These commands work only with selected objects at design time. In code, you can use the **ZOrder** method to move forms to the front or back of other forms and objects to the front or back of other objects.

Visual Basic Menus

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbidxVisualBasicMenusC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbidxVisualBasicMenusS"}

File Menu

Edit Menu

View Menu

Project Menu

Format Menu

Debug Menu

Run Menu

Tools Menu

Add-Ins Menu

Window Menu

Help Menu

File Menu

{ewc HLP95EN.DLL,DYNALINK,"See Also":"cmdFileMenuC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"cmdFileMenuS"}

New Project

Open Project

Add Project

Remove Project

Save Project/Save Project Group

Save Project As/Save Project Group As

Save

Save As

Print

Print Setup

Make Project

Make Project Group

File 1, 2, 3, 4

Exit

Edit Menu

{ewc HLP95EN.DLL,DYNALINK,"See Also":"cmdEditMenuC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"cmdEditMenuS"}

Undo

Redo

Cut

Copy

Paste

Paste Link

Delete

Select All

Find

Find Next

Replace

Indent

Outdent

Insert File

List Properties/Methods

List Constants

Quick Info

Parameter Info

Complete Word

Bookmarks

View Menu

{ewc HLP95EN.DLL,DYNALINK,"See Also":"cmdViewMenuC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"cmdViewMenuS"}

Code

Object

Definition

Last Position

Object Browser

Immediate Window

Locals Window

Watch Window

Call Stack

Project Explorer

Properties Window

Form Layout Window

Property Pages

Toolbox

Color Palette

Toolbars

Run Menu

{ewc HLP95EN.DLL,DYNALINK,"See Also":"cmdRunMenuC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"cmdRunMenuS"}

Start

Start With Full Compile

Break

End

Restart

Tools Menu

{ewc HLP95EN.DLL,DYNALINK,"See Also":"cmdToolsMenuC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"cmdToolsMenuS"}

Add Procedure

Procedure Attributes

Menu Editor

Options

Add-Ins Menu

{ewc HLP95EN.DLL,DYNALINK,"See Also":"cmdAddInsMenuC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"cmdAddInsMenuS"}

Visual Data Manager

Add-In Manager

Format Menu

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdFormatMenuC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdFormatMenuS"}

Align

Make Same Size

Size to Grid

Horizontal Spacing

Vertical Spacing

Center in Form

Order

Lock Controls

Debug Menu

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnDebugMenuC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnDebugMenuS"}

Step Into

Step Over

Step Out

Run To Cursor

Add Watch

Edit Watch

Quick Watch

Toggle Breakpoint

Clear All Breakpoints

Set Next Statement

Show Next Statement

Project Menu

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdProjectMenuC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdProjectMenuS"}

Add Form

Add MDI Form

Add Module

Add Class Module

Add User Control

Add Property Page

Add User Document

Add ActiveX Designer

Add File

Remove <Item>

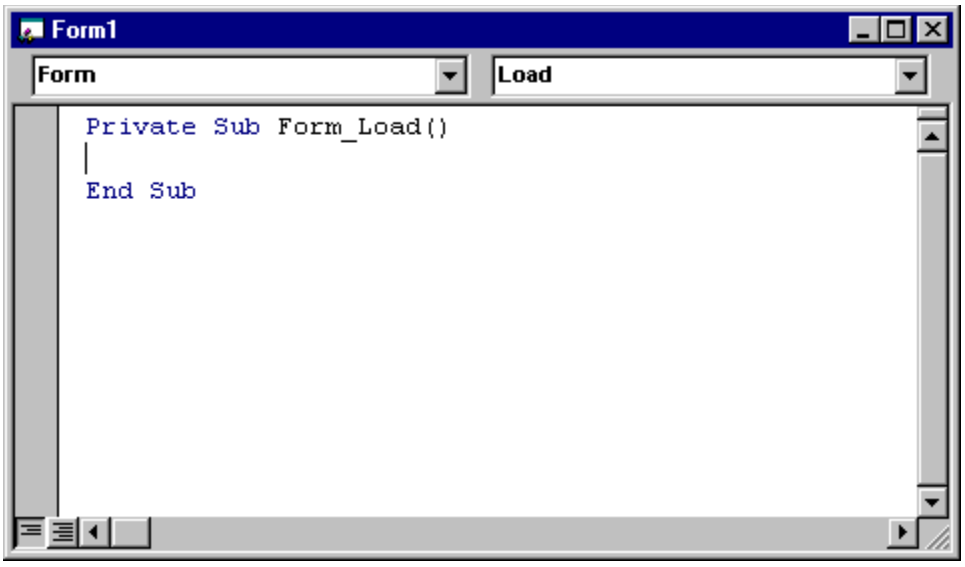
References

Components

<Project> Properties

Code Window

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnCodeWindowC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnCodeWindowS"}



Use the Code window to write, display, and edit Visual Basic code. You can open as many Code windows as you have modules, so you can easily view the code in different forms or modules, and copy and paste between them.

You can open a Code window from:

- The Project window, by selecting a form or module, and choosing the View Code button.
- A Form window, by double-clicking a control or form, choosing Code from the View menu, or pressing F7.

You can drag selected text to:

- A different location in the current Code window.
- Another Code window.
- The Immediate and Watch windows.
- The Recycle Bin.

Window Elements

Object Box

Displays the name of the selected object. Click the arrow to the right of the list box to display a list of all objects associated with the form.

Procedures/Events Box

Lists all the events recognized by Visual Basic for a form or control displayed in the Object box. When you select an event, the event procedure associated with that event name is displayed in the Code window.

If (General) is displayed in the Object box, the Procedure box lists any declarations and all of the general procedures that have been created for the form. If you are editing module code, the Procedure box lists all of the general procedures in the module. In either case, the procedure you select in the Procedure box is displayed in the Code window.

All the procedures in a module appear in a single, scrollable list that is sorted alphabetically by name. Selecting a procedure using the drop down list boxes at the top of the Code window moves the cursor to the first line of code in the procedure you select.

Split Bar

Dragging this bar down, splits the Code window into two horizontal panes, each of which scrolls separately. You can then view different parts of your code at the same time. The information that appears in the Object box and Procedures/Events box applies to the code in the pane that has the focus. Dragging the bar to the top or the bottom of the window or double-clicking the bar closes a pane.

Margin Indicator Bar

A gray area on the left side of the Code window where margin indicators are displayed.

Procedure View Icon

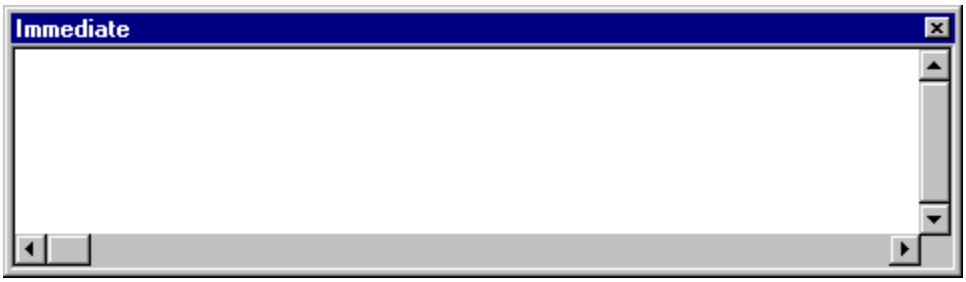
Displays the selected procedure. Only one procedure at a time is displayed in the Code window.

Full Module View Icon

Displays the entire code in the module.

Immediate Window

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnImmediateWindowC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnImmediateWindowS"}



Automatically opens in break mode and is empty.

You can:

- Type or paste a line of code and press ENTER to run it.
- Copy and paste the code from the Immediate window into the Code window but you cannot save code in the Immediate window.

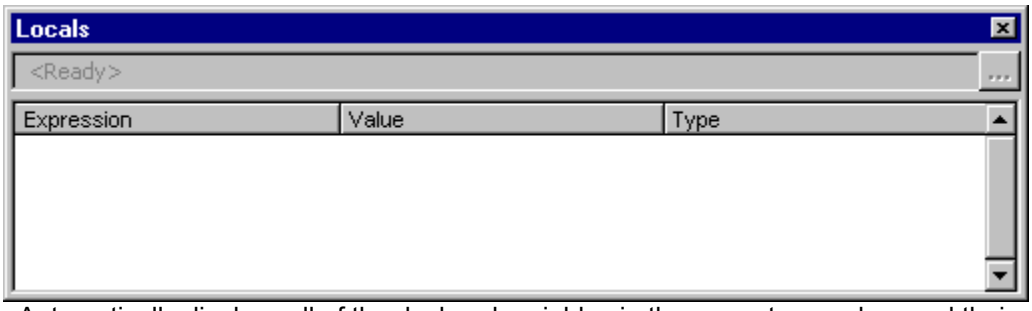
The Immediate window can be dragged and positioned anywhere on your screen unless you have made it a dockable window from the Docking Tab of the Options dialog box.

You can close the window by clicking the Close box. If the Close box is not visible, double-click the Title bar to make the Close box visible, then click it.

Note In break mode, a statement in the Immediate window is executed in the context or scope that is displayed in the Procedure box. For example, if you type **Print** *variablename*, your output is the value of a local variable. This is the same as if the **Print** method had occurred in the procedure you were executing when the program halted.

Locals Window

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnLocalsWindowC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnLocalsWindowS"}



Automatically displays all of the declared variables in the current procedure and their values.

When the Locals window is visible, it is automatically updated every time there is a change from Run to Break mode or you navigate in the stack display

You can:

- Resize the column headers by dragging the border to the right or the left.
- Close the window by clicking the Close box. If the Close box is not visible, double-click the Title bar to make the Close box visible, then click it.

Window Elements

Calls Stack Button Opens the Call Stack dialog box which lists the procedures in the call stack.

Expression Lists the name of the variables.

The first variable in the list is a special module variable and can be expanded to display all module level variables in the current module. For a class module, the system variable <Me>is defined. For standard modules, the first variable is the <name of the current module>. Global variables and variables in other projects are not accessible from the Locals window.

You cannot edit data in this column.

Value List the value of the variable.

When you click on a value in the Value column, the cursor changes to an I-beam. You can edit a value and then press ENTER, the UP ARROW key, the DOWN ARROW key, TAB, SHIFT+TAB, or click on the screen to validate the change. If the value is illegal, the Edit field remains active and the value is highlighted. A message box describing the error also appears. Cancel a change by pressing ESC.

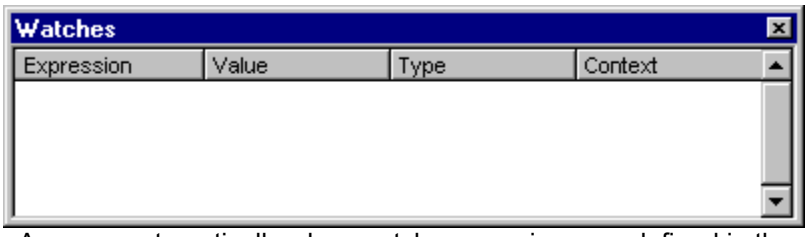
All numeric variables must have a value listed. String variables can have an empty Value list.

Variables that contain subvariables can be expanded and collapsed. Collapsed variables do not display a value but each subvariable does. The expand icon, ⊞ and the collapse icon, ⊟ appear to the left of the variable.

Type Lists the variable type. You cannot edit data in this column.

Watch Window

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnWatchWindowC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnWatchWindowS"}



Appears automatically when watch expressions are defined in the project.

You can:

- Change the size of the column headers by dragging its border to the right to make it larger or to the left to make it smaller.
- Drag a selected variable to the Immediate window or the Watch window
- Close the window by clicking the Close box. If the Close box is not visible, double-click the Title bar to make the Close box visible, then click it.

Window Elements

Expression Lists the watch expression with the Watch icon,  on the left.

Value List the value of the expression at the time of the transition to break mode.

You can edit a value and then press ENTER, the UP ARROW key, the DOWN ARROW key, TAB, SHIFT+TAB, or click somewhere on the screen to validate the change. If the value is illegal, the Edit field remains active and the value is highlighted. A message box describing the error also appears. Cancel a change by pressing ESC.

Type Lists the expression type.

Context Lists the context of the watch expression.

If the context of the expression isn't in scope when going to break mode, the current value isn't displayed.

You can close the window by clicking the Close box. If the Close box is not visible, double-click the Title bar to make the Close box visible, then click it.

Margin Indicators

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vargnMarginWidgetsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vargnMarginWidgetsS"}

Visual Basic contains margin indicators that provide visual cues to certain actions during the editing of your code. If more than one action occurs on a line, the appropriate indicators also appear.

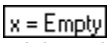
The margin indicators appear in the Margin Indicator bar on left side of the Code window.

You can turn the Margin Indicator Bar on and off in the Editor Format tab of the Options dialog box.

Margin Indicator	Margin Indicator Name	Description
	Breakpoint	Indicates that you have set a breakpoint using the Toggle Breakpoint command on the Debug menu. You can toggle the breakpoint by placing your mouse pointer in the margin indicator region and clicking.
	Current line of execution	Indicates the line of code that will be executed next. You can drag this margin indicator to a new location within any running code module. If you drag the Current line of execution margin indicator to any non-valid region or line, nothing happens and the indicator returns to the original location.
	Bookmark	Indicates the location of a bookmark set using the Toggle Bookmark command on the Edit menu.
	Call Stack Marker	Indicates lines that are currently in the call stack. The Call Stack Marker indicator appears only in break mode.

Data Tips Window

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vargnAutoVariableWindowC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vargnAutoVariableWindowS"}



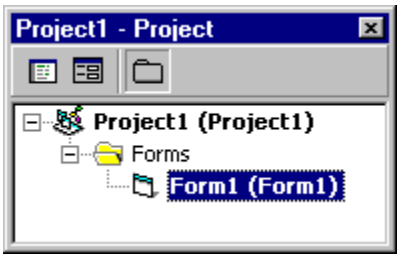
A box that displays the value of the variable over which your cursor is placed.

For example, if you set `x=5` and then set breakpoint before the end of your code, when you run your code and place your cursor over the “x”, the value 5 appears in the Data Tips window.

Available in break mode.

Project Explorer

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnProjectWindowC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnProjectWindowS"}



Displays a hierarchical list of the projects and all of the items contained in a project.

Window Elements

View Code

Displays the Code window so you can write and edit code associated with the selected item.

View Object

Displays the Object window for the selected item, an existing form, module, ActiveX object, or user control.

Toggle Folders

Hides and shows the object folders while still showing the individual items contained within them.

List window

Lists the all loaded projects and the items included in each project.

Project

The project and items contained within it.

- Forms

All .frm files associated with the project.

- Modules

All .bas modules for the project.

- Class Modules

All .cls files for the project.

- User Controls

All user controls for the project.

- User Documents

All document objects, .dob files, in the project.

- Property Pages

All property pages, .pag files, in the project.

- Related Documents

  Lists all documents to which you want a pointer. The path to the document is stored rather than the document itself. When you click View Object, Visual Basic searches the registry for the document type and executes the appropriate open command. You can place any valid document type in the project.

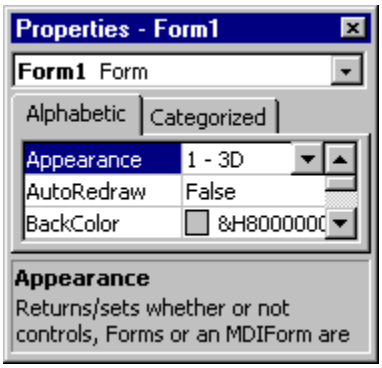
- Resources

  Lists all of the resources you have in your project.

Note A check mark to the left of a filename in the Project Explorer indicates that the file is checked out of a version control project, and currently has read/write status. The check mark is only displayed if you are connected to a version control program, such as Microsoft Visual SourceSafe, which is included with the Enterprise Edition of Visual Basic.

Properties Window

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnPropertiesWindowC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnPropertiesWindowS"}



Lists the design-time properties for selected objects and their current settings. You can change these properties at design time. When you select multiple controls, the Properties window contains a list of the properties common to all the selected controls.

Window Elements

Object Box

Lists the currently selected object. Only objects from the active form are visible. If you select multiple objects, the properties common to the objects and their settings, based on the first object selected, appear on the Properties List tabs.

Properties List Tabs

- **Alphabetic Tab —** Alphabetically lists all properties for the selected object that can be changed at design time, as well as their current settings. You can change the property setting by selecting the property name and typing or selecting the new setting.
- **Categorized Tab —** Lists all properties for the selected object by category. For example, **BackColor**, **Caption**, and **ForeColor** are in the Appearance category. You can collapse the list so that you see the categories or you can expand a category to see the properties. When you expand or collapse the list, you see a plus (+) icon or minus (-) icon to the left of the category name.

Description Pane

Shows the property type and a short description of the property. You can turn the description of the property off and on using the Toggle Status command on the shortcut menu. You can move through the list of descriptions by pressing the ARROW keys.

Data Types Displayed in Locals Window

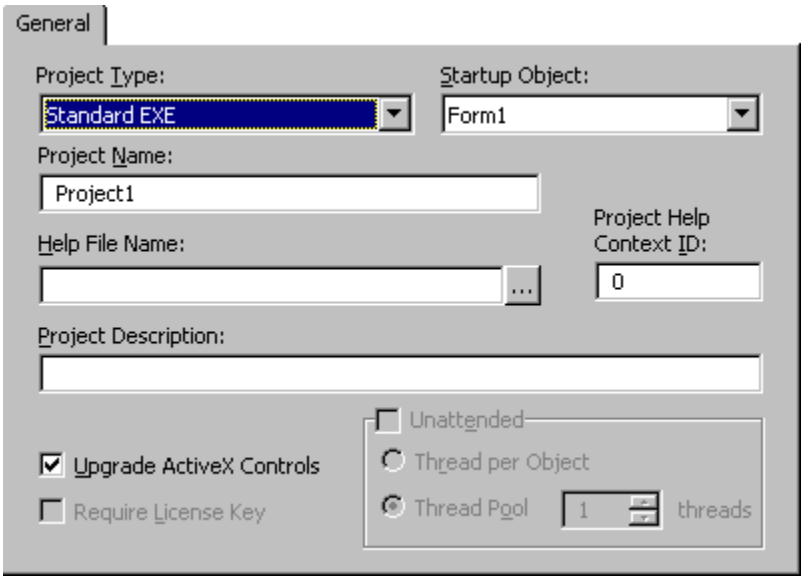
{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamscDataTypesInLocalsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamscDataTypesInLocalsS"}

The following is a list of data types and their default state when they are displayed in the Locals window.

Data Type	Default State
Currency	Value is displayed.
Double	Value is displayed.
Integer	Value is displayed
Long	Value is displayed.
Single	Value is displayed.
String	Value is displayed.
UDT	No value is displayed. Non-editable. Collapsed.
Variant	Value is displayed.
Object	No value is displayed. Non-editable. Collapsed.
Decimal	Value is displayed.
Array	No value is displayed. Non-editable. Collapsed.

General Tab (Project Properties Dialog Box)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgGeneralTabProjectSettingsDialogC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgGeneralTabProjectSettingsDialogS"}



Specifies the settings for the current Visual Basic project. The name of the project is displayed in the title bar.

Tab Options

Project Type Sets the project type. You can change any project type to another project type. When you change a project type, all associated changes are made automatically. If items are invalid in the new project type, they remain part of the project but you cannot add new instances. If you have properties in the current project that are not valid in the new project, you will get a message suggesting that you do not make the change.

Startup Object Sets which form or Sub/Main in the current project runs first.

Project Name Identifies your component in the Windows Registry and the Object Browser. It is important that it has a unique name.

The project name is the name of the *type library* for your component. The type library, or TypeLib, contains the description of the objects and interfaces provided by your component.

It is also used to qualify the names of classes. A combination of project name and class name is sometimes referred to as a *fully qualified class name*, or as a *programmatic ID*. The fully qualified class name may be required to correctly identify an object as belonging to your component.

Help File Name Displays the name of the Help file associated with the project.

Project Help Context ID Lists the context ID for the specific Help topic to be called when the user selects the ⊞ button while the application's object library is selected in the Object Browser.

Project Description Sets the descriptive text that is displayed in the Description pane at the bottom of the Object Browser.

You can use the Project Description as the description for ActiveX components. If you are creating an ActiveX control, it is the text that appears in the Components dialog box.

If you are creating an ActiveX DLL or ActiveX EXE, this text appears in the References dialog box.

Project Load Allows you to set the following conditions when a project is loaded.

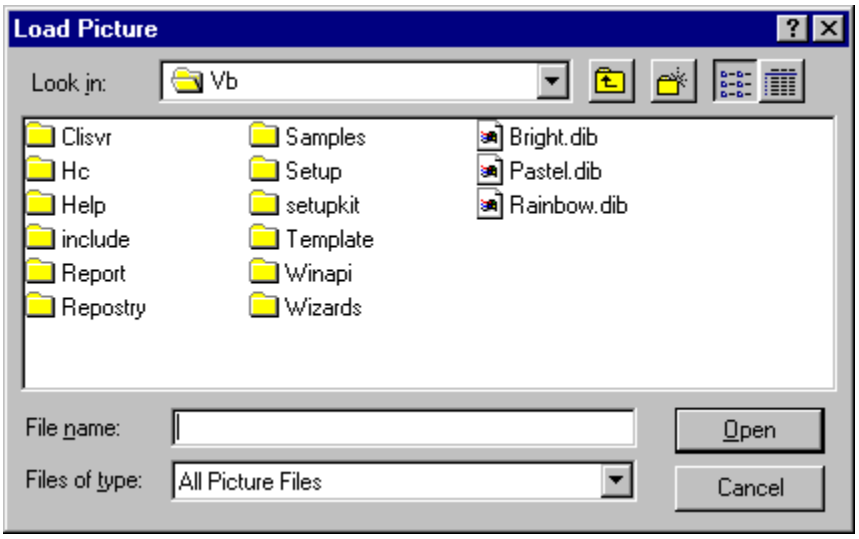
- Upgrade ActiveX Controls — Enables upgrading of the ActiveX controls.
- Require License Key — Enables licensing for a project that produces ActiveX components (automation servers, user controls, or ActiveX controls). A Visual Basic license file (*.vbl) will be created when you build the file. The *.vbl must be registered on the user's machine for the components to be used. The SetUp Wizard registers the *.vbl file.

Unattended Execution Indicates that the project is intended to run without user interaction. Unattended projects have no interface elements. Any runtime functions such as messages that normally result in user interaction are written to an event log.

- Thread per Object — Indicates that each instance of a class marked as Multiuse in the **Instancing** property will be created on a new and distinct thread. Each thread has a unique copy of all global variables and objects, and will not interfere with any other thread.
- Thread Pool — Indicates that each instance of a class marked as Multiuse in the **Instancing** property will be created on a thread from the thread pool. The choice of thread is determined in a round robin fashion. Each thread has a unique copy of all global variables, but multiple instances reside on a given thread and can potentially interfere with each other.
- Number of threads — Determines the maximum number of threads created for the thread pool. When a Multiuse class is instantiated, threads are created as needed up to the number set here. After the maximum number is reached, Visual Basic begins assigning new instances to existing threads.

Load Picture, Load Icon Dialog Boxes

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vadlgLoadPictureLoadIconDialogsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vadlgLoadPictureLoadIconDialogsS"}



Loads an existing picture or icon file.

Dialog Box Options

Look in

Select the location of the picture you want to open.

Up One Level

Shows a list of folders or drives one level above the current folder.

Create New Folder

Creates a new folder.

List

Shows the folders or files in a list format that includes the icon and its name.

Details

Shows the folders or files in a list that includes the icon and name, its size, type, and the date and time it was last modified.

File name

Select or type the name of a picture file you want to open.

List files of type

Select a file type. Files of the selected type appear in the File name list box.

- All Picture Files (*.bmp, *.dib, *.wmf, *.emf, *.ico, *.cur)—Lists all picture files.

- Bitmaps(*.bmp, *.dib)—Lists bitmaps and DIBs.
- Metafiles (*.wmf, *.wmf)—Lists Microsoft Windows Metafiles.
- Icons (*.ico, *.cur)—Lists the icons.
- All Files (*.*)—Lists files of all types.

Open

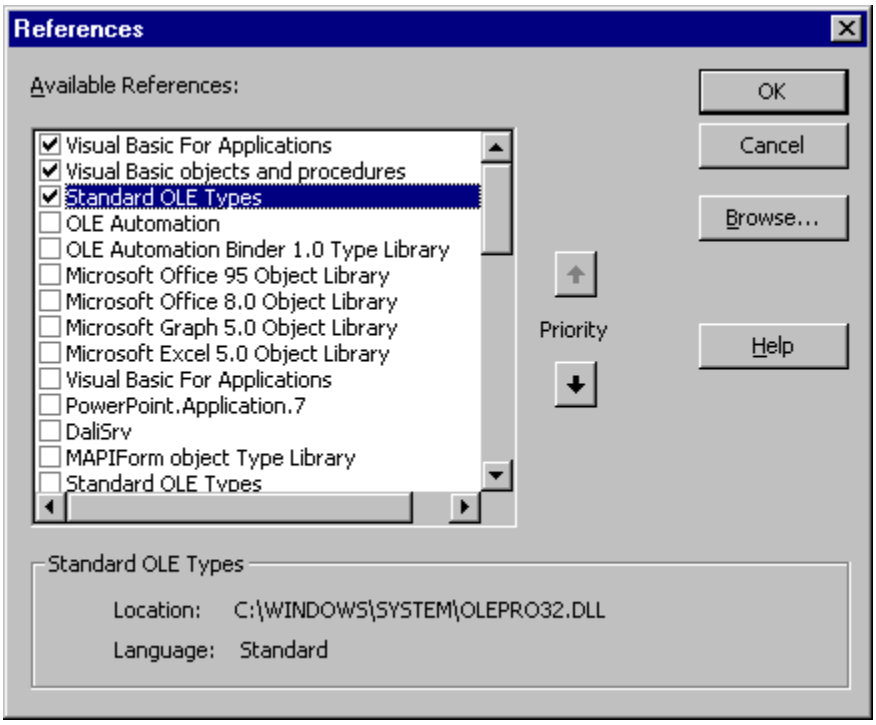
Opens the selected file.

Cancel

Closes the dialog box without loading a picture file.

References Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnAddReferencesC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnAddReferencesS"}



Allows you to select another application's objects that you want available in your code by setting a reference to that application's object library.

Dialog Box Options

Available References Lists the references available to your project.

- After you set a reference to an object library by selecting the check box next to its name, you can find a specific object and its methods and properties in the Object Browser.
- If you are not using any objects in a referenced library, you should clear the check box for that reference to minimize the number of object references Visual Basic must resolve, thus reducing the time it takes your project to compile. You can't remove a reference for an item that is used in your project.
- If you remove a reference to an object that your are currently using in your project, you will receive an error the next time you refer to that object.
- References not in use are listed alphabetically.

Note You can't remove the "Visual Basic for Applications" and "Visual Basic objects and procedures" references, because they are necessary for running Visual Basic.

Priority Buttons Moves references up,  , and down,  , on the list. When you refer to an object in code, Visual Basic searches each referenced object selected in the References dialog box in the order the referenced objects are displayed. If two referenced objects use the same name, Visual Basic uses the definition provided by the referenced object listed higher in the Available References box.

Result Displays the name and path of the reference selected in the Available References box, as

well as the language version.

Browse Displays the Add Reference dialog box so that you can search other directories for and add references to the Available Resources box for the following types:

- Type Libraries (*.olb, *.tlb, *.dll)
- Executable Files (*.exe, *.dll)
- ActiveX Controls (*.ocx)
- All Files (*.*)

The Add References dialog box is the Open common dialog box.

Break Command (Run Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdBreakC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdBreakS"}

Stops execution of a program while it's running and switches to break mode. Any statement being executed when you choose this command is displayed in the Code window with ⊞ in the left margin if you checked Margin Indicator Bar in the Editor Format tab of the Options dialog box. If the application is waiting for events in the idle loop (no statement is being executed), no statement is highlighted until an event occurs.

Some editing changes made in break mode may require you to restart your program for the changes to take effect.

This command is available only at run time.

Toolbar button:  Keyboard shortcut: CTRL+BREAK.

Code Window Keyboard Shortcuts

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdShortcutC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdShortcutS"}

You can use the following shortcut keys to access commands in the Code window.

Description	Shortcut Keys
View Code window	F7
View Object Browser	F2
Find	CTRL+F
Replace	CTRL+H
Find Next	F3
Find Previous	SHIFT+F3
Next procedure	CTRL+DOWN ARROW
Previous procedure	CTRL+UP ARROW
View definition	SHIFT+F2
Shift one screen down	CTRL+PAGE DOWN
Shift one screen up	CTRL+PAGE UP
Go to last position	CTRL+SHIFT+F2
Beginning of module	CTRL+HOME
End of module	CTRL+END
Move one word to right	CTRL+RIGHT ARROW
Move one word to left	CTRL+LEFT ARROW
Move to end of line	END
Move to beginning of line	HOME
Undo	CTRL+Z
Delete current line	CTRL+Y
Delete to end of word	CTRL+DELETE
Indent	TAB
Outdent	SHIFT+TAB
Clear all breakpoints	CTRL+SHIFT+F9
View shortcut menu	SHIFT+F10

Code Window General Use Keys

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vacmdCodeWindowGeneralUseKeysC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vacmdCodeWindowGeneralUseKeysS"}

Use these key combinations in the Code window:

Press	To
F1	Get context-sensitive Help on functions, statements, methods, properties, or events.
F2	Display the Object Browser.
F9	Set or remove a breakpoint.
CTRL+SHIFT+F9	Clear all breakpoints.
F5	Run an application (or continue running, if in break mode).
F8	Execute code one line at a time (single step).
SHIFT+F8	Execute code one procedure at a time (procedure step).
CTRL+BREAK	Stop running a Visual Basic application.
HOME	Move the cursor to the beginning of text in a line.
END	Move the cursor to the end of text in a line.
Double-click on the split bar	Delete the split bar.
CTRL+J	Turn on List Properties/Methods.
CTRL+SHIFT+J	Turn on List Constants.
CTRL+I	Turn on Quick Info.
CTRL+SHIFT+I	Turn on Parameter Info.
CTRL+SPACEBAR	Turn on Complete Word.
SHIFT+F10	View shortcut menu.
SHIFT+F5	Restart an application from the beginning.
ALT+F5	Runs the error handler code or returns the error to the calling procedure. Does not affect the setting for error trapping on the General tab of the Options dialog box.
ALT+F8	Steps into the error handler or returns the error to the calling procedure. Does not affect the setting for error trapping on the General tab of the Options dialog box.

Code Window Navigation Keys

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vakbdCodeWindowNavigationC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vakbdCodeWindowNavigationS"}

Use these key combinations to navigate in the Code window:

Press	To
CTRL+F7	Move the insertion point into the Object box.
SHIFT+F2	Go to the definition of the selected procedure.
CTRL+DOWN ARROW	Display the next procedure.
CTRL+UP ARROW	Display the previous procedure.
PAGE DOWN	Page down through the procedures in your code.
PAGE UP	Page up through the procedures in your code.
CTRL+SHIFT+F2	Go back to the last position in your code.
CTRL+HOME	Go to the beginning of the module.
CTRL+END	Go to the end of the module.
CTRL+RIGHT ARROW	Go one word to the right.
CTRL+LEFT ARROW	Go one word to the left.
END	Go to the end of the line.
HOME	Go to the beginning of the line.
CTRL+PAGE DOWN	Go to the bottom of the current procedure.
CTRL+PAGE UP	Go to the top of the current procedure.
F6	Switch between Code window panes (when the window is split).

Code Editing Keys

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vakbdCodeEditingKeysC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vakbdCodeEditingKeysS"}

Use these key combinations to edit code in the Code window:

Press	To
CTRL+C	Copy the selected text to the Clipboard.
CTRL+X	Cut the selected text to the Clipboard.
DELETE or DEL	Delete the selected text without placing it on the Clipboard.
CTRL+V	Paste the Clipboard contents at the insertion point.
CTRL+Z	Undo the last editing action in the current line.
CTRL+Y	Cut the current line to the Clipboard.
CTRL+DELETE	Delete to the end of the word.
CTRL+BACKSPACE	Delete to the beginning of the word.
F3	Find Next: repeat text search down through your code. If no text search has been done, the Find dialog box is displayed.
SHIFT+F3	Find Previous: repeat text search up through your code. If no text search has been done, the Find dialog box is displayed.
SHIFT+TAB	Remove indent.
CTRL+N	Insert a blank line above the current line.

Menu Shortcut Keys Available in the Code Window

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vakbdMenuShortcutKeysInCodeWindowC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vakbdMenuShortcutKeysInCodeWindowS"}

Use these key combinations for menu shortcuts in the Code window:

Press	To
CTRL+P	Print
CTRL+Z	Undo
CTRL+V	Paste
DEL or DELETE	Delete
CTRL+F	Find
F3	Find Next
CTRL+H	Replace
TAB	Indent
SHIFT+TAB	Outdent
CTRL+J	List Properties/Methods
CTRL+SHIFT+J	List Constants
CTRL+I	Quick Info
CTRL+SHIFT+I	Parameter Info
CTRL+SPACEBAR	Complete Word
SHIFT+F2	Definition
CTRL+SHIFT+F2	Last Position
F2	Object Browser
CTRL+G	Immediate Window
CTRL+R	Project Explorer
F4	Properties Window
F8	Step Into
SHIFT+F8	Step Over
CTRL+F8	Run To Cursor
SHIFT+F9	Quick Watch
F9	Toggle Breakpoint.
CTRL+SHIFT+F9	Clear All Breakpoints
F5	Start
CTRL+BREAK	Break
SHIFT+F10	Shortcut menu
CTRL+N	New Project
CTRL+O	Open Project
CTRL+S	Save Form
CTRL+A	Save Form As
CTRL+F5	Start with Full Compile
SHIFT+F4	Property Pages
CTRL+D	Add File

Immediate Window Keyboard Shortcuts

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vakbdImmediateWindowKeysC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vakbdImmediateWindowKeysS"}

Use these key combinations in the Immediate window:

Press	To
ENTER	Run a line of selected code.
CTRL+C	Copy the selected text to the Clipboard.
CTRL+V	Paste the Clipboard contents at the insertion point.
CTRL+X	Cut the selected text to the Clipboard.
CTRL+L	Display Call Stack dialog box (break mode only).
F5	Continue running an application.
F8	Execute code one line at a time (single step).
SHIFT+F8	Execute code one procedure at a time (procedure step).
DELETE or DEL	Delete the selected text without placing it on the Clipboard.
F2	Display the Object Browser.
CTRL+ENTER	Insert carriage return.
CTRL+HOME	Move the cursor to the top of the Immediate window.
CTRL+END	Move the cursor to the end of the Immediate window.
SHIFT+F10	View shortcut menu.
ALT+F5	Runs the error handler code or returns the error to the calling procedure. Does not affect the setting for error trapping on the General tab of the Options dialog box.
ALT+F8	Steps into the error handler or returns the error to the calling procedure. Does not affect the setting for error trapping on the General tab of the Options dialog box.
SHIFT+F5	Restart an application.
F6	Switch between the Immediate window and the Watch window (if visible).

Watch Window Keyboard Shortcuts

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vakbdWatchWindowKeyboardShortcutC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vakbdWatchWindowKeyboardShortcutS"}

Use these key combinations in the Watch window:

Press	To
SHIFT+ENTER	Display the selected watch expression.
CTRL+W	Display Edit Watch dialog box.
ENTER	Expands or collapses the selected watch value if it has a plus (+) or minus (-) to the left of it.
F2	Display the Object Browser.
SHIFT+F10	View shortcut menu.
F6	Switch between the Watch window and the Immediate window.

Properties Window Keyboard Shortcuts

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vakbdPropertiesWindowsKeysC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vakbdPropertiesWindowsKeysS"}

Use these key combinations when the Property list has the focus in the Properties window:

Press	To
PAGE DOWN	Move down through the Property list,.
PAGE UP	Move up through the Property list.
DOWN ARROW	Move down through the Property list, one property at a time.
UP ARROW	Move up through the Property list, one property at a time.
RIGHT ARROW	Move down through the Property list, one property at a time.
LEFT ARROW	Move up through the Property list, one property at a time.
END	Move to the last property in the list.
HOME	Move to the first property in the list.
ALT+F6	Switch between the last two active windows.
TAB	Move the insertion point among the property, properties settings box, and the Object box.
CTRL+SHIFT+ALPHA	Move to the next property in the list that begins with the alpha character.
Double-Click	Cycle through settings of enumerated properties, or switch focus to the settings box for other property types.

Use these key combinations when the settings box for a property has the focus in the Properties window:

Press	To
CTRL+Z	Undo the last editing action in the current line.
CTRL+C	Copy the selected text to the Clipboard.
CTRL+X	Cut the selected text to the Clipboard.
DEL or DELETE	Delete the selected text without placing it on the Clipboard.
CTRL+V	Paste the Clipboard contents at the insertion point.
SHIFT+TAB	Switch focus between the Object box and the active Properties tab.
TAB	Move the focus among the Object box, the active Properties tab, a property, and the property value.
ESC	Cancel the property change.
CTRL+SHIFT+ALPHA	Move to the next property that begins with the alpha character, in the list.

Use these key combinations when a property has enumerated values and its settings box has the focus in the Properties window:

Press	To
ALT+DOWN ARROW	Open the settings box list.
ALT+UP ARROW	Close the settings box list.

Use these key combinations when you set the **BackColor**, **ForeColor**, **FillColor**, or **BorderColor** properties, and the settings box has the focus in the Properties window:

Press	To
ALT+DOWN ARROW	Display the Color palette.
ALT+UP ARROW	Close the Color palette.
DEL	Reset to (None).

Use these key combinations when you set the **Icon** or **Picture** properties, and the settings box has the focus in the Properties window:

Press	To
ALT+DOWN ARROW	Display the Load Icon or Load Picture dialog box.
ALT+UP ARROW	Display the Load Icon or Load Picture dialog box.
DEL	Reset to (None).

Project Explorer Keyboard Shortcuts

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vakbdProjectWindowC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vakbdProjectWindowS"}

Use these key combinations in the Project Explorer:

Press	To
ENTER	Open the selected file from the list, or expand and collapse the list to show its subentries.
SHIFT+ENTER	Open the Code window for the selected file.
F7	Open the Code window for the selected file.
SHIFT+F10	View shortcut menu.
HOME	Select the first file in the list.
END	Select the last file in the list.
RIGHT ARROW	Expands a list and then selects a subentry in the list each time you press it.
LEFT ARROW	Selects a subentry in the list and then moves up the list each time you press it until the subentry list collapses into a folder.
UP ARROW	Moves up the list one entry at a time.
DOWN ARROW	Moves down the list one entry at a time.

Global Keys

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbkbdGlobalC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbkbdGlobalS"}

Use these key combinations in all Visual Basic windows:

Press	To
F5	Run an application.
F8	Execute code one line at a time.
SHIFT+F8	Execute statements one line at a time without stepping into procedure calls.
CTRL+BREAK	Stop running a Visual Basic application.
CTRL+G	Displays the Immediate window.
ALT+F5	Runs the error handler code or returns the error to the calling procedure. Does not affect the setting for error trapping on the General tab of the Options dialog box.
ALT+F8	Steps into the error handler or returns the error to the calling procedure. Does not affect the setting for error trapping on the General tab of the Options dialog box.
SHIFT+F5	Restart an application from the beginning after an interruption.
CTRL+TAB	Switch between windows.

Use these key combinations in all Windows-based applications:

Press	To
F1	Open Help.
ALT+F6	Toggle between the last two active windows.
ALT+F4	(Visual Basic) Close the active window; if all windows are closed, close Visual Basic.
CTRL+C	Copy the selection to the Clipboard.
CTRL+X	Cut the selection to the Clipboard.
CTRL+V	Paste the Clipboard selection.
CTRL+Z	Undo the last edit.
SHIFT+F10	View shortcut menu.

Options Command (Tools Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdOptionsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdOptionsS"}

Displays the Options dialog box, from which you can choose a tab to set attributes of the Visual Basic programming environment. Available only at design time.

Tabs

- Editor
- Editor Format
- General
- Docking
- Environment
- Advanced

New Project Command (File Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdNewProjectC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdNewProjectS"}

Displays the New Project dialog box where you choose the type of project you want to create.

If there is currently another project open when you create a new project, you will be prompted to save your work.

Available only at design time.

Color Palette Command (View Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdColorPaletteSpinC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdColorPaletteSpinS"}

Displays or activates the Color palette, which enables you to change a form or control's colors and set up a custom color scheme.

Available only at design time.

Open Project Command (File Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdOpenProjectC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdOpenProjectS"}

Closes the current project or group project, if one is loaded, and opens an existing project or group of projects.

You can open as many projects as your system resources permit.

Toolbar shortcut:  . Keyboard shortcut: CTRL+O.

Note If you are using an integrated source code control program, such as Microsoft SourceSafe (available with the Enterprise Edition), and have checked the project files out of the version control project, when you close a project (before opening a new one), a dialog may prompt you to check in the files you have checked out.

When you open a project in Visual Basic, a dialog may prompt you to get or check out the Visual Basic project files.

Save and Save As Commands (File Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdSaveFileC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdSaveFileS"}

Saves the current or selected project files to the location you specified. The Save command displays the Save File As dialog box if this is the first time the file is being saved.

The dialog box enables you to specify a new name for a component. Available only at design time.

Keyboard shortcut: CTRL+S (Save).

Note If you are using an integrated source code control program, such as Microsoft SourceSafe (available with the Enterprise Edition), when you save a file with the Save As command, a dialog may prompt you to add the file to the version control project. You can add a file to a SourceSafe project only if the Visual Basic project file (.vbp) is checked out.

When you use Save As to rename a file that is checked out of a source code control program, ensure that the source code control program also renames the file in the version control project.

Dialog Box Options

Save in

Select the folder where you want to store the file.

Up One Level

Shows a list of folders or drives one level above the current folder.

Create New Folder

Creates a new folder.

Details

Shows the folder or documents in a list that includes the folder or document icon and name, its size (documents only), type, and the date and time it was last modified.

List

Shows the folders or documents in a list format that includes the folder or document icon and its name.

File name

Unless you specify a filename extension, Visual Basic automatically adds the default filename extension.

Save as type

Select a file type from the list:

- All Files (*.*) — Lists files of all types.
- Form Files (*.frm) — Lists all forms and MDI forms.
- Class Files (*.cls) — Lists all class modules.
- Basic Files (*.bas) — Lists all standard modules.
- Property Page Files (*pag) — List all property pages.
- User Defined Control Files (*.ctl) — Lists all user controls.

- Document Object Files (*.dob) — Lists all document objects.

Save
Saves the file under the specified name.

Cancel
Closes the dialog box without saving the file.

Save Project, Save Project As, Save Project Group, and Save Project Group As Commands (File Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdSaveProjectC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdSaveProjectAsS"}

Save Project Saves the current project and all its components.

Save Project As The Save Project command displays the Save Project As dialog box if this is the first time the project is being saved.

Save Project Group The Save Project command changes to Save Project Group if you add a project to the project group.

Save Project Group As The Save Project As command displays the Save Project Group As dialog box if this is the first time the project group is being saved.

Available only at design time.

Toolbar shortcut:  .

Dialog Box Options

Save In

Select the location where you want to store the project file.

Up One Level

Shows a list of folders or drives one level above the current folder.

Create New Folder

Creates a new folder.

List

Shows the folders or documents in a list format that includes the folder or document icon and its name.

Details

Shows the folder or documents in a list that includes the folder or document icon and name, its size (documents only), type, and the date and time it was last modified.

List

Shows the list of folders or documents.

File name

Give your project a name. To save a project with a new name, or in a different location, type a new filename. To save a project with an existing filename, select the name in a list or type the current name. When you choose Save, Visual Basic asks if you want to overwrite the existing file

Save as type

Select a file type from the list; the default is Project (*.vbp). Files of the selected type will appear in the File Name list box.

Save

Saves the project group under the specified name.

Cancel

Closes the dialog box without saving the project.

Add File Command (Project Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdAddFileC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdAddFileS"}

Adds an existing file to the current project. You can share files between projects. If you add a file to a project, you are simply including a reference to the existing file in the project; you are not adding a copy of the file to the project. Therefore, if you make changes to the file and save it, your changes will affect any project that includes the file.

Available at design time only.

Keyboard shortcut: CTRL+D.

Note Use the Components command on the Project menu to add ActiveX controls and insertable objects to your project's Toolbox.

Note If you have connected to a source code control program, such as Microsoft SourceSafe (available with the Enterprise Edition), when you add a file to a Visual Basic project, a dialog may prompt you to add the file in the version control project.

With Microsoft SourceSafe, if the .vbp file is checked out, the references to the newly added file(s) are automatically added. If the .vbp file isn't checked out, when you add a file, the .vbp file will not contain a reference to the new file, and thus, it will not load the next time you open your project.

Dialog Box Options

Look in

Select the location from which you want to add the file.

Up One Level

Shows a list of folders or drives one level above the current folder.

Create New Folder

Creates a new folder.

List

Shows the folders or documents in a list format that includes the folder or document icon and its name.

Details

Shows the folder or documents in a list that includes the folder or document icon and name, its size (documents only), type, and the date and time it was last modified.

File name

Select or type the name of a file you want to open. This box lists documents with the filename extension selected in the List Files of Type box.

List files of type

Select a file type from the list:

- VB Files (*.frm; *.ctl, *.pag, *.bas; *.cls; *.res) – Lists all valid Visual Basic file types.
- Form Files (*.frm) — Lists all form modules.
- User Defined Control Files (*.clt) — Lists all user defined controls.
- Property Page Files (*.pag) — Lists all property pages.

- User Document Files (.dob) — Lists all user documents.
- Basic Files (*.bas) — Lists all standard modules.
- Class Files (*.cls) — Lists all class modules.
- Resource Files (*.res) — Lists all resource files. There can be only one resource file in a single project.
- All Files (*.*) — Lists all files.

Add

Adds the file to the project.

Cancel

Closes the dialog box without adding the file to the project.

Remove Project Command (File Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdRemoveFileC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdRemoveFileS"}

Removes the selected project from the currently open project group.

If there are pending changes to the project, you will be prompted to save them and then the project will be closed and removed from the project group.

If you remove a project with a reference to any of the EXEs or other projects in the group, Visual Basic automatically switches the reference to the binary version of the component, if one is available. If a binary version is not available, the reference is marked as missing.

Exit Command (File Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdExitC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdExitS"}

Closes the current project and quits Visual Basic. If you try to quit Visual Basic before saving changes to your work, you'll be prompted to save your work first.

Note When you exit Visual Basic, you may be prompted to check in the Visual Basic files to the source code control project if you are using an integrated source code control program, such as Microsoft SourceSafe (available with the Enterprise Edition), and you have any files checked out.

Keyboard shortcut: ALT+Q.

Lock Controls Command (Format Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdLockC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdLockS"}

Locks all controls on the form in their current positions so you don't inadvertently move them once they are in the desired location. Because the Lock Controls command works on a form-by-form basis, it locks controls only on the selected form; controls on other forms are untouched. Visual Basic keeps track of which forms have controls locked in position and which don't. When the controls on the form are locked, the Lock Controls command toolbar button appears dimmed.

Toolbar shortcut:  .

Start With Full Compile (Run Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdStartWithFullCompileC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdStartWithFullCompileS"}

Performs a full compile on the project.

The application is fully compiled before starting, regardless of the current setting of the Compile on Demand and Background Compile options on the General tab of the Options dialog box. Choosing Start With Full Compile does not alter the Compile on Demand or Background compile settings.

Use Start With Full Compile to start ActiveX server projects. Starting an ActiveX server project with the Start command when Compile on Demand checked may result in compile-time errors being detected after you have test applications running. If correcting an error requires restarting the ActiveX server, your test applications may be left holding invalid object references.

This command is available only at design time.

Keyboard shortcut: CTRL+F5.

Add-In Manager Command (Tools Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdAddInManagerC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdAddInManagerS"}

Displays the Add-In Manager dialog box, which you use to load and unload add-in which extend the Visual Basic development environment.

Property Pages Command (View Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdPropertyPageC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdPropertyPageS"}

Displays the property pages for a user control which to change a control's properties at design time.

Keyboard shortcut: SHIFT+F4.

Add User Control Command (Project Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdCustomControlC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdCustomControls"}

Displays the Add User Control dialog box so you can insert a new or existing user control into your active project.

Toolbar shortcut:  .

End Command (Run Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdRunEndC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdRunEndS"}

Stops running the program and returns to design time. This command is available at run time and in break mode.

Toolbar shortcut:  .

Menu Editor Command (Tools Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdMenuEditorC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdMenuEditorS"}

Displays the Menu Editor dialog box.

Use the Menu Editor command to create custom menus for your application ,and to define some of their properties. Available only at design time.

Toolbar shortcut:  . Keyboard shortcut: CTRL+E.

Add MDI Form Command (Project Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdMDIFormC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdMDIFormS"}

Displays the Add MDI Form dialog box so you can insert a new or existing MDI form into your active project. A project can have only one MDI form. This command is unavailable if a project already has an MDI form.

Only available at design time.

Toolbar shortcut:  .

Add Form Command (Project Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdFormInsertC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdFormInsertS"}

Displays the Add Form dialog box so you can insert a new or existing form into your active project.

Only available at design time.

Toolbar shortcut:  .

Start Command (Run Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdStartC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdStartS"}

Runs the application marked as the Start Up project in the Project Explorer. All forms being designed are closed, variables are initialized, and the startup form (if any) is loaded. You cannot run control projects at design time. DLL project types can be run by another EXE project creating an instance of them, and group projects cannot be run unless an EXE project exists in the group.

The first EXE project added to a project group is automatically set as the Start Up project unless you change it using the Set as Start Up command on the shortcut menu.

Available only at design time. The Start command becomes the Break command at run time, and begins executing the code from the current execution line margin indicator, ⊞. It becomes the Continue command in break mode.

Note If you are working on a sub project and choose the Start command, the main project changes to run mode. The sub project is compiled but is not executed until the main project makes a call to it.

Toolbar shortcut:  . Keyboard shortcut: F5.

Make <Project> Command (File Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdMakeProjectC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdMakeProjectS"}

Opens the Make Project dialog box so that you can build one or more projects contained in the project group into an executable file – EXE, DLL, or OCX.

Note You can have only one main project. If you want to debug other projects in the design environment, you must add the .vbp files to the group project.

Make Project Group Command (File Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdMakeAllC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdMakeAllS"}

Creates an separate executable file for each project in the group you select.

This is the same as using the **/make** flag in the command line and specifying a .vbg file.

Add User Document Command (Project Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdDocumentObjectInsertC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdDocumentObjectInsertS"}

Displays the Add User Document dialog box so you can insert a new or existing User Document into your active project.

Only available if you have an ActiveX EXE or ActiveX DLL project.

Toolbar shortcut:  .

Continue Command (Run Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdContinueVB5C;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdContinueVB5S"}

Resumes execution of a program that is in break mode.

Replaces the Start command when the program is in break mode.

References Command (Project Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdReferencesCommandProjectMenuC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdReferencesCommandProjectMenuS"}

Displays the References dialog box. This dialog box allows you to add an object library or type library, or project reference to your project. This makes another application's objects available in your code. Once a reference is set, the referenced objects are displayed in the Object Browser.

You can also add references to other loaded and saved projects. If a project has not been saved, it appears as "UNSAVED: <ProjectName>" and you will be unable to make a reference to it.

Only available at design time.

Toolbar shortcut:  .

Restart Command (Run Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdRestartC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdRestartS"}

Restarts your application after any kind of interruption. An interruption can be caused by run-time errors, a Stop statement, a breakpoint in your code, choosing the Break command, and a Break expression changing or becoming true.

This command is available only in break mode.

Keyboard shortcut: SHIFT+F5.

Help Menu

{ewc HLP95EN.DLL,DYNALINK,"See Also":"cmdHelpMenuC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"cmdHelpMenuS"}

Microsoft Visual Basic Help Topics

Runs Help and displays the Visual Basic Table of Contents.

Toolbar shortcut:  .

Search Reference Index

Displays Help's Search dialog box so you can quickly find the reference information you need.

Toolbar shortcut:  .

Search Master Index

Displays the viewer to access documentation about Visual Basic. Appears only if you installed Visual Basic Books Online.

Toolbar shortcut:  .

Obtaining Technical Support

Runs Help and displays information about Microsoft Product Support Services.

Toolbar shortcut:  .

Microsoft on the Web

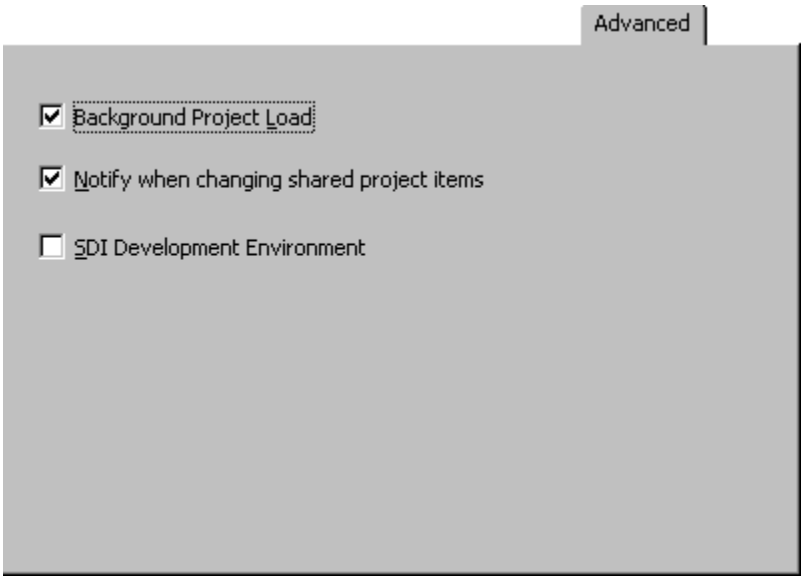
Displays a menu with Internet sites.

About Microsoft Visual Basic

Displays a dialog box with the version number, a copyright notice, and the amount of available memory.

Advanced Tab (Options Dialog Box)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnAdvancedOptionsTabC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnAdvancedOptionsTabS"}



Allows you to specify the settings for various advanced features as they apply to the current Visual Basic project.

Tab Options

Background Project Load Determines whether code is loaded in the background, returning control to the developer more quickly.

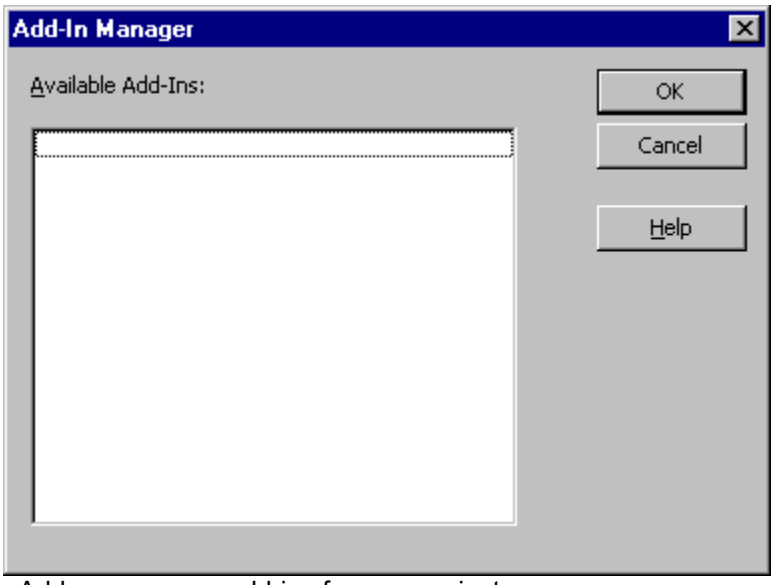
Notify when changing shared project items Determines whether you will be notified when you change a shared project item such as a form or module and try to save it.

Several projects can share the same items. Shared items are loaded into memory and each project has its own copy. If you change a shared item in one project, the other projects retain the copy of the item that was loaded until you save the projects. Then, the last project you save determines what the shared file is. When this option is selected, you are asked if you want to synchronize all of the copies of the item before you save.

SDI Development Environment When selected changes the development environment from a multiple document interface (MDI) to a single document interface (SDI). When you select this option, the SDI appears every time you restart Visual Basic until you clear this option.

Add-In Manager Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnAddInManagerC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnAddInManagerS"}



Adds or removes add-ins for your project.

When you select an add-in, it is loaded every time you restart Visual Basic until you clear it.

Dialog Box Options

Available Add-Ins box Lists the:

- Currently running add-ins
- Add-ins listed in vbaddin.ini

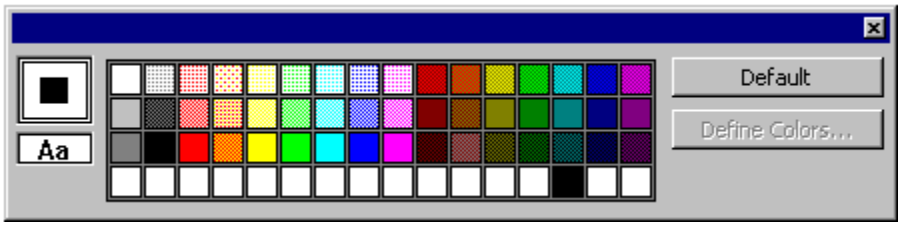
All currently running add-ins are checked in the list box, including those running in other instances of Visual Basic. It is up to the developer of the add-in to ensure that the add-in is registered in the system registry when you install the add-in; otherwise, it won't appear in this box.

The check box to the left of each name indicates whether that add-in is used by the current project.

OK Updates Vbaddin.ini with your selections and unloads the deselected add-ins or loads the selected add-ins.

Color Palette

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnColorPaletteC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnColorPaletteS"}



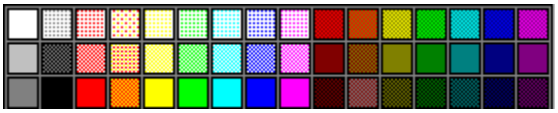
Changes the colors of a form or control and sets up a custom color scheme.

The Color Palette consists of:

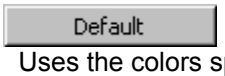
Displays the currently selected foreground and background colors for the form or control.

Displays the currently selected foreground and background colors for any text in the form or control.

Note If Microsoft Windows doesn't display the text and background colors that you've selected, it may be because one of the colors you've selected is dithered (a color comprised of up to three different colored pixels). When displaying text in Microsoft Windows, the text and the area immediately behind it must be solid colors. If you choose a dithered color for either the text or the background color, the nearest solid color is used.

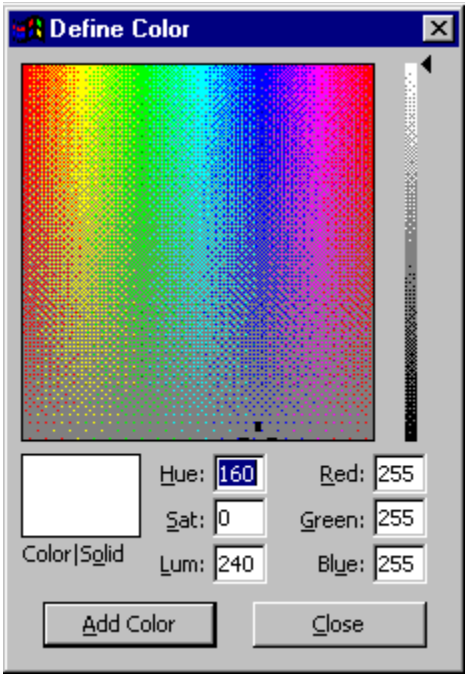


Displays a set of colors.



Uses the colors specified in the Control Panel.

Displays or sets custom colors.

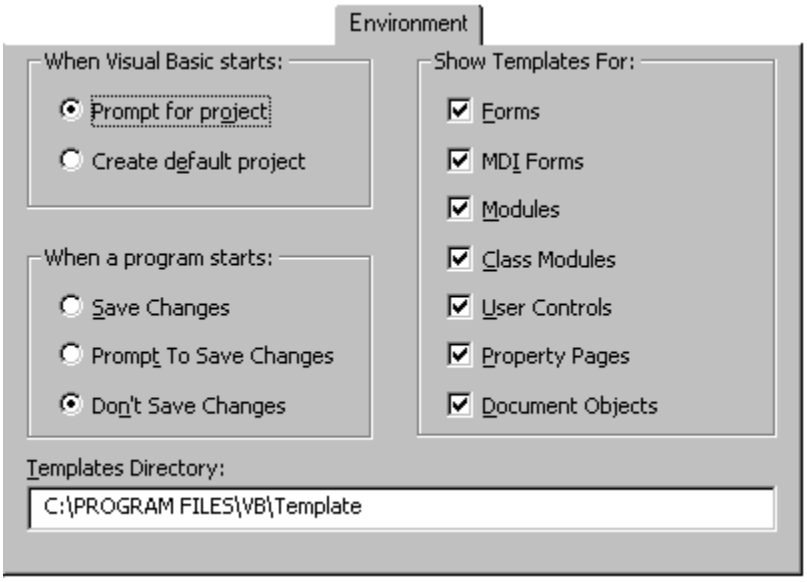


Displays the Define Color dialog box so you can create custom colors.

Note You can also set the colors for forms and controls with the **BackColor** or **ForeColor** property.

Environment Tab (Options Dialog Box)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnEnvironmentOptionsC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnEnvironmentOptionsS"}



Specifies the attributes of your Visual Basic development environment.

Changes made in this dialog box are saved in the registry file, and are loaded every time you restart Visual Basic.

Tab Options

When Visual Basic starts

- Prompt for project — Asks you for the project you want to open each time you start Visual Basic.
- Create default project — Creates a default executable (EXE) project that opens each time you start Visual Basic.

When a program starts

- Save Changes — Automatically saves the changes without prompting you when you make changes to a project and then press F5 to run it or choose the Start command from the Run menu. If you have a new file, the Save As common dialog box appears so you can give a name and location for your project.
- Prompt To Save Changes — Always displays a dialog box asking if you want to save the changes to your project when you press F5 to run it or choose the Start command from the Run menu. If you select Yes, the Save As common dialog box appears so you can give a name and location for your project. If you select No, Visual Basic runs the project using the memory image, but does not save any changes.
- Don't Save Changes — When you run your project, Visual Basic runs the memory version and does not save the changes.

Show Templates For: Allows you to determine which templates you want visible in the Add <item> dialog box when you add an item to a project. If cleared, a blank form appears when you choose the Add <item> command.

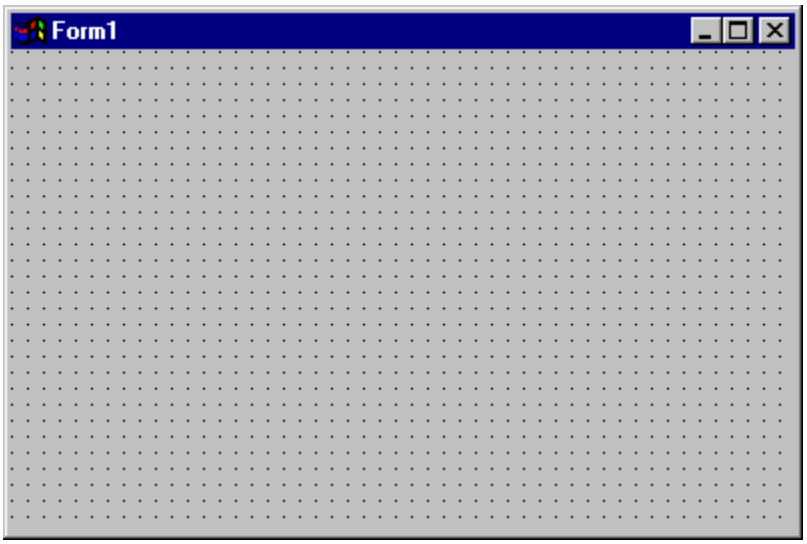
- Forms
- MDI Forms
- Modules
- Class Modules

- User Controls
- Property Pages
- User Documents

Templates Directory Lists the location of the template files.

Form Window

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnFormWindowC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnFormWindowS"}



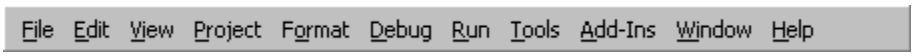
Allows you to create the windows, dialog boxes, and controls in your application. You draw and view controls on a form.

While you are designing a form:

- Each form window has a Maximize, Minimize, and Close button.
- You can create either fixed or movable forms. The form you design will have the same features at design time and at run time unless you specify otherwise in the form's properties.
- Use the buttons in the Toolbox to draw controls on the form.
- Use the Form Layout Window command from the View menu to preview the layout of your form on a screen.

Menu Bar

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnMenuBarC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnMenuBarS"}



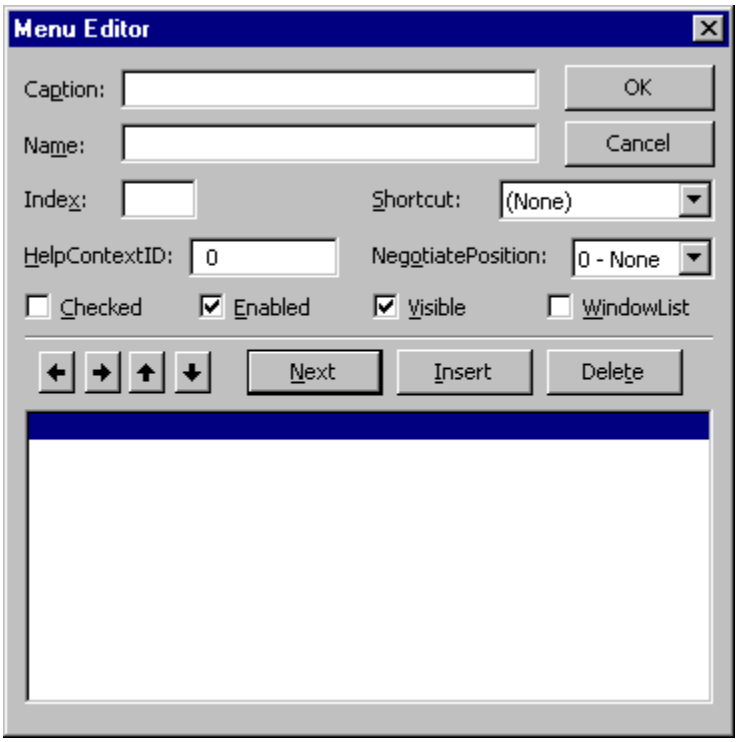
Lists the menus that you can use in the active window.

The Visual Basic menu bar contains the names of the menus you can use in the active window. You can modify the menu bar using the Commands tab of the Customize dialog box.

Note You cannot delete the menu bar.

Menu Editor

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnMenuEditorC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnMenuEditorS"}



Allows you to create custom menus for your application and to define their properties.

Dialog Box Options

Caption Allows you to enter the menu or command name that you want to appear on your menu bar or in a menu.

If you want to create a separator bar in your menu, type a single hyphen (-) in the Caption box.

To give the user keyboard access to a menu item, insert an ampersand (&) before a letter. At run time, this letter is underlined (the ampersand is not visible), and the user can access the menu or command by pressing ALT and the letter. If you need an ampersand to show in the menu, put two consecutive ampersands in the caption.

Name Allows you to enter a control name for the menu item. A control name is an identifier used only to access the menu item in code; it doesn't appear in a menu.

Index Allows you to assign a numeric value that determines the control's position within a control array. This position isn't related to the screen position.

Shortcut Allows you to select a shortcut key for each command.

HelpContextID Allows you to assign a unique numeric value for the context ID. This value is used to find the appropriate Help topic in the Help file identified by the **HelpFile** property.

NegotiatePosition Allows you to select the menu's **NegotiatePosition** property. This property determines whether and how the menu appears in a container form.

Checked Allows you to have a check mark appear initially at the left of a menu item. It is generally used to indicate whether a toggle option is turned on or off.

Enabled Allows you to select whether you want the menu item to respond to events, or clear if you want the item to be unavailable and appear dimmed.

Visible Allows you to have the menu item appear on the menu.

WindowList Determines if the menu control contains a list of open MDI child forms in an MDI application.

Right Arrow

Moves the selected menu down one level each time you click it. You can create up to four levels of submenus.

Left Arrow

Moves the selected menu up one level each time you click it. You can create up to four levels of submenus.

Up Arrow

Moves the selected menu item up one position within the same menu level each time you click it.

Down Arrow

Moves the selected menu item down one position within the same menu level each time you click it.

Menu List A list box that displays a hierarchical list of menu items. Submenu items are indented to indicate their hierarchical position or level.

Next Moves selection to the next line.

Insert Inserts a line in the list box above the currently selected line

Delete Deletes the currently selected line.

OK Closes the Menu Editor and applies all changes to the last form you selected. The menu is available at design time, but selecting a menu at design time opens the Code window for that menu's Click event rather than executing any event code.

Cancel Closes the Menu Editor and cancels all changes.

Toolbox

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnToolboxC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnToolboxS"}

Displays the standard Visual Basic controls plus any ActiveX controls and insertable objects you have added to your project. You can:

- Display ToolTips for the Toolbox buttons by selecting the Show ToolTips option in the General tab of the Options dialog box.
- Customize the Toolbox by adding tabs to it.

  When you add a tab, a Pointer is always available on it.
- Customize the General tab or a custom tab by adding controls using the Components command from the Project menu.

Standard Toolbox Controls

Pointer

The only item in the Toolbox that doesn't draw a control. When you select the pointer, you can only resize or move a control that has already been drawn on a form.

PictureBox

Displays graphical images (either decorative or active), as a container that receives output from graphics methods, or as a container for other controls.

Label

Allows you to have text that you don't want the user to change, such as a caption under a graphic.

TextBox

Holds text that the user can either enter or change.

Frame

Allows you to create a graphical or functional grouping for controls. To group controls, draw the Frame first, and then draw controls inside the frame.

CommandButton

Creates a button the user can choose to carry out a command.

CheckBox

Creates a box that the user can easily choose to indicate if something is true or false, or to display multiple choices when the user can choose more than one.

OptionButton

Allows you to display multiple choices from which the user can choose only one.

ComboBox

Allows you to draw a combination list box and text box. The user can either choose an item from the list or enter a value in the text box.

ListBox

Used to display a list of items from which the user can choose one. The list can be scrolled if it has more items than can be displayed at one time.

HScrollBar (horizontal scroll bar)

Provides a graphical tool for quickly navigating through a long list of items or a large amount of information, for indicating the current position on a scale, or as an input device or indicator of speed or quantity.

VScrollBar (vertical scroll bar)

Provides a graphical tool for quickly navigating through a long list of items or a large amount of information, for indicating the current position on a scale, or as an input device or indicator of speed or quantity.

Timer

Generates timer events at set intervals. This control is invisible at run time.

DriveListBox

Displays valid disk drives.

DirListBox (directory list box)

Displays directories and paths.

FileListBox

Displays a list of files.

Shape

Allows you to draw a variety of shapes on your form at design time. You can choose a rectangle, rounded rectangle, square, rounded square, oval, or circle.

Line

Used to draw a variety of line styles on your form at design time.

Image

Displays a graphical image from a bitmap, icon, or metafile on your form. Images displayed in an **Image** control can only be decorative and use fewer resources than a **PictureBox**.

Data

Provides access to data in databases through bound controls on your form.

OLE

Allows you to link and embed objects from other applications in your Visual Basic application.

Code Command (View Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdCodeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdCodeS"}

Displays or activates the Code window for a currently selected object.

Toolbar shortcut: ⊞. Keyboard shortcut: F7.

Last Position Command (View Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdLastPositionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdLastPositionS"}

Allows you to quickly navigate to a previous location in your code. Enabled only if you edited code or made a Definition command call and only when the Code window is displayed. Visual Basic only keeps track of the last 8 lines that were accessed or edited. Available at design time and in break mode.

Keyboard shortcut: CTRL+SHIFT+F2.

Object Browser Command (View Menu)

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

Displays the Object Browser, which lists the object libraries, the type libraries, classes, methods, properties, events, and constants you can use in code, as well as the modules and procedures you defined for your project. Not available at run time.

Toolbar shortcut:  . Keyboard shortcut: F2.

Properties Window Command (View Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdPropertiesWindowC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdPropertiesWindowS"}

Displays the Properties window, which lists the design-time properties for a selected form, control, class, user control, property page, user document, or menu. Available only at design time.

Toolbar shortcut:  . Keyboard shortcut: F4.

Toolbox Command (View Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdToolBoxC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdToolboxS"}

Displays or hides the Toolbox, which contains the controls and insertable objects (such as a Microsoft Excel Chart) currently available to your application. Available only at design time.

Toolbar shortcut:  .

Call Stack Command (View Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdCallsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdCallsS"}

Displays the Calls dialog box, which lists the procedure calls in the application that have started but are not completed. Available only in break mode.

When Visual Basic is executing the code in a procedure, that procedure is added to a list of active procedure calls. If that procedure then calls another procedure, there are two procedures on the list of active procedure calls. Each time a procedure calls another **Sub**, **Function**, or **Property** procedure, it is added to the list. Each procedure is removed from the list as execution is returned to the calling procedure. Procedures called from the Immediate window are also added to the calls list.

You can also display the Calls dialog box by clicking the Calls button (...) next to the Procedure box in the Locals window.

Toolbar shortcut:  Keyboard shortcut: CTRL+L.

Object Command (View Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdObjectViewC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdObjectViewS"}

Displays the active item.

Only available at design time and when the cursor is on a valid object.

Toolbar shortcut: ⊞. Keyboard shortcut: SHIFT+F7.

Immediate Window Command (View Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdImmediateWindowC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdImmediateWindowS"}

Displays the Immediate window and displays information resulting from debugging statements in your code or from commands typed directly into the window.

Available in break mode.

Use the Immediate window to:

- Test problematic or newly written code.
- Query or change the value of a variable while running an application. While execution is halted, assign the variable a new value as you would in code.
- Query or change a property value while running an application.
- Call procedures as you would in code.
- View debugging output while the program is running.

Toolbar shortcut:  . Keyboard shortcut: CTRL+G.

Locals Window Command (View Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdLocalsWindowC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdLocalsWindowS"}

Displays the Locals window and automatically displays all of the variables in the current stack and their values.

The Locals window is automatically updated every time you change from run time to break mode and every time the stack context changes.

Toolbar shortcut:  .

Watch Window Command (View Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdWatchWindowC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdWatchWindowS"}

Displays the Watch window and displays the current watch expressions. The Watch window appears automatically if watch expressions are defined in the project.

If the context of the expression isn't in scope when going to break mode, the current value isn't displayed.

Toolbar shortcut: ⊞.

Definition Command (View Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdProcedureDefinitionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdProcedureDefinitionS"}

Displays the location in the Code window where the variable or procedure under the pointer is defined. If the definition is in a referenced library, it is displayed in the Object Browser.

Keyboard shortcut: SHIFT+F2.

Toolbars Command (View Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdCommandBarsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdCommandBarsS"}

Lists the toolbars that are built into Visual Basic and the Customize command. You can:

- Toggle the toolbars on and off
- Drag the toolbars to different locations on you desktop.

Debug Displays the Debug toolbar which contains buttons for common debugging tasks.

Edit Displays the Edit toolbar which contains buttons for common editing tasks.

Form Editor Displays the Form Editor toolbar which contains buttons specific to editing a form.

Standard Displays the Standard toolbar which is the default toolbar.

Customize Displays the Customize dialog box where you can customize or create toolbars and your menu bar.

Project Explorer Command (View Menu)

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

Displays the Project Explorer, which displays a hierarchical list of the currently open projects and their contents.

The Project Explorer is a navigational and management tool only. You cannot build an application from the Project Explorer. You can close a project by choosing the Remove Project command from the shortcut menu.

Toolbar shortcut:  . Keyboard shortcut: CTRL+R.

Window Menu Commands

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdWindowsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdWindowsS"}

Split

Toggles splitting the Code window in half horizontally. Only available when the Code window is active.

Toolbar shortcut: 

Tile Horizontally

Tiles the Object and Code windows and the Object Browser horizontally when you are in MDI mode.

Toolbar shortcut: 

Tile Vertically

Tiles the Object and Code windows and the Object Browser vertically when you are in MDI mode.

Toolbar shortcut: 

Cascade

Rearranges the Object and Code windows and the Object Browser in your project so they overlap in a cascade. Available only in MDI mode.

Toolbar shortcut: 

Arrange Icons

Arranges the icons of the windows you have minimized, neatly at the bottom left of the window.

Window List

Lists all open form windows

Command Line Arguments

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

Determines whether Visual Basic complies and runs a program, compiles and makes an executable (.exe) file or ActiveX DLL (.dll), or sets the argument part of the command line returned by the Command function.

Syntax

vb5[.exe] [[{**/run | /r**} *projectname*] [[{**/d** *compileconst*}] {**/make** | **/m** } *projectname*] [{**/makedll | /l**} *projectname*] {/**cmd** *argument* **| /c** *argument*}][{**/runexit**} *projectname*][{**/m**} or {**/runexit**} *projectname* **/out** *filename*}][{**/m**}][**/sdi**] or [**/mdi**]

The parts of the command line switch syntax are:

Argument	Description
projectname	The name of your project (.vbp) file.
/run or **/r**	Tells Visual Basic to compile and run *projectname* using the arguments stored in the Command Line Arguments field of the Make tab of the Project Properties dialog box. You can run more than one project using this command. Replace *projectname* with *projectgroupname*.
/make or **/m**	Tells Visual Basic to compile *projectname* and make an executable (.exe) file, using the existing settings of the **Path**, **EXEName**, and **Title** properties of the APP object. You can compile and make an executable (.exe) file from more than one project using this command. Replace of the *projectname* with *projectgroupname*.
/**makedll** or **/l**	Tells Visual Basic to compile *projectname* and make an in-process ActiveX server (.dll) file from it.
/**d** or **/D**	Tells Visual Basic which values to use for conditional compilation constants when making an .EXE with the /**make** switch or an ActiveX DLL with the /**makedll** switch.
compileconst	The names and values of conditional compilation constants used in the project file.
/**cmd** or **/c**	Puts argument in the Command Line Arguments field in the Make tab of the Project Properties dialog box. When used, this must be the last switch on the command line.
/runexit	Tells Visual Basic to run *projectname*. If for any reason the file is changed in the process of running, all changes are ignored and no dialog appears on exit to design mode.
filename	The name of the file to receive errors when you build an executable using the /**m** or **/runexit** option.
/out	Allows you to specify a file to receive errors when you build using the **/m** or **/runexit** option. The first error encountered is placed in this file with other status information. If you do not use the /out option, command line build errors are displayed in a message box. This option is useful if you are building multiple projects.
/?	Lists the available Command Line arguments.
/sdi	Changes the Visual Basic environment to SDI (Single Document Interface) mode. Visual Basic remains in SDI mode until you change it. You can change to MDI mode by using the **/mdi** argument or by clearing the SDI Development Environment option in the Advanced tab of the Options dialog box.
/mdi	Opens Visual Basic in MDI (Multiple Document Interface) mode. Visual

Basic remains in MDI mode until you change it. You can change to SDI mode by using the /**sdi** argument or by selecting the SDI Development Environment option in the Advanced tab of the Options dialog box. MDI mode is the default.

When used, these arguments must be in a command line to run Visual Basic. For example, you might use them in the Run dialog box from the Run command of the Windows 95 Start Menu. Here is a valid command line that runs Visual Basic, loads a specific project, and runs it:

```
C:\vb5.exe /r MyProj.vbp
```

Visual Basic Design Time File Extensions

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

Visual Basic has a number of file extensions that are specific to design time. The currently used file extensions are:

Extension	Used For
.bas	Basic module
.cls	Class module
.ctl	User Control file
.ctx	User Control Binary file
.dca	Active Designer Cache
.dep	Setup Wizard dependency file
.dob	User Document form file
.dox	User Document binary form file
.dsr	Active Designer file
.dsx	Active Designer binary file
.frm	Form file
.frx	Binary Form file
.log	Log file for load errors
.oca	Control Typelib Cache
.pag	Property Page file
.pgx	Binary Property Page file
.res	Resource file
.swt	Visual Basic Set Up Wizard Template file
.tlb	Remote Automation Typelib files
.vbg	Visual Basic group project
.vbl	User Control Licensing file
.vbp	Visual Basic project
.vbr	Remote Automation Registration files
.vbw	Visual Basic Project workspace
.vbz	Wizard launch file

Visual Basic Run Time File Extensions

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

Extension	Used For
.dll	In-process ActiveX server
.exe	EXE file, Out-of-process server
.ocx	ActiveX control

Project Types

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

Project type is a new property added to each project. Project type determines:

- The type of component build when you choose the Make Project command from the File menu.
- Whether the ActiveX DLL restrictions are in effect for the project.
- Whether a project can contain creatable public classes.
- The valid options for the Startup Object.

The following table lists the project types and their characteristics and the characteristics of a Project Group.

Project Type	**Standard EXE**	**ActiveX EXE**	**ActiveX DLL**	**ActiveX Control**	**Project Group**
Type	EXE	EXE	DLL	DLL	Group
Default Extension	.exe	.exe	.dll	.ocx	.vbg
DLL Restrictions	No	No	Yes	Yes	No
Creatable Public Class	No	Yes	Yes	Yes	No
Start Mode	Stand Alone	OLE Server	Cannot run unless a subproject in the main project. Automatically starts running when needed.	Cannot run unless a subproject in the main project. Automatically starts running when needed.	Cannot run unless a subproject in the main project. Automatically starts running when needed.
Default Startup Object	Form1	None	None	None	None
Allowable Objects	MDI Form Form Module Class Module User Control Property Page ActiveX Designer	Form Module Class Module User Control Property Page User Document ActiveX Designer	Form Module Class Module User Control Property Page User Document ActiveX Designer	Form Module Class Module User Control Property Page ActiveX Designer	

Templates

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

Templates are items that are saved in a specific directory structure in the \vb directory. They provide a way for you to re-use components and code, and to add them to projects.

When you choose the New Project command on the File menu, any projects found in the \template\projects directory appear in the New Project dialog box.

You can set the path for your template directory, or turn templates off, using the Environment tab of the Options dialog box.

Standard Toolbar

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdStandardCommandBarVB5C;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdStandardCommandBarVB5S"}

Contains buttons that are shortcuts to some commonly used menu items.

You can click a toolbar button once to carry out the action represented by that button. You can select the Show ToolTips option in the General tab of the Options dialog box if you want to display ToolTips for the toolbar buttons.

Toolbar Buttons

Add Project Displays a submenu listing the types of projects you can add to the currently open project group. The icon changes to the last project type you added. The default is the Standard EXE.

— Standard EXE

— ActiveX EXE

— ActiveX DLL

— ActiveX Control

Add <item> Displays a submenu listing the items you can add to your active project. The icon changes to the last object you added. The default is the Form.

— Form

— MDI Form

— Module

— Class Module

— User Control

— Property Page

— User Document

Add ActiveX Designer

Add File

Show Menu

Displays the Menu Editor dialog box.

Open Project

Closes the current project and project group, if one is loaded, and displays an existing group project and it accompanying projects.

Save Project

Saves the current project and all its components–forms and modules.

Cut

Removes the selected control or text and places it on the Clipboard.

Copy

Copies the selected control or text onto the Clipboard.

Paste

Inserts the contents of the Clipboard at the current location.

Find

Searches for the specified text in a search range specified in the Find dialog box.

Undo

Reverses the last editing action, such as typing text in the Code window or deleting controls.

Redo

Restores the last text editing if no other actions have occurred since the last Undo.

Start

Runs the application starting with the Startup Object identified on General tab of the Project Properties dialog box.

Break

Stops execution of a program while it's running and switches to break mode.

End

Stops running the program and returns to design time

Project Explorer

Displays the Project Explorer, which displays a hierarchical list of the currently open projects and their contents.

Properties Window

Displays the Properties window so you can view the properties of the selected control.

Form Layout Window

Displays the Form Layout window where you can preview position your form within the window.

Object Browser

Displays the Object Browser, which lists the object libraries, the type library, classes, methods, properties, events, and constants you can use in code, as well as the modules and procedures you defined for your project.

Toolbox

Displays the Toolbox, which contains the controls and insertable objects (such as a Microsoft Excel Chart) currently available to your application.

Edit Toolbar

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdEditCommandBarC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdEditCommandBarS"}

Contains buttons that are shortcuts to some commonly used menu items frequently used when editing code.

You can click a toolbar button once to carry out the action represented by that button. You can select the Show ToolTips option in the General tab of the Options dialog box if you want to display ToolTips for the toolbar buttons.

Toolbar Buttons

List Properties/Methods

Displays a box in the code window that contains the properties and methods available for the selected object.

List Constants

Displays a box in the Code window that contains the constants that are valid choices for the property you typed and that precede the equals sign (=).

Quick Info

Provides the syntax for a function, method, or procedure based on the location of your pointer within the name of the function, method, or procedure.

Parameter Info

Shows a popup in the Code window that contains information about the parameters of the function in which the pointer is located.

Complete Word

Accepts the characters that Visual Basic automatically adds to the word you are typing.

Indent

Shifts all lines in the selection to the next tab stop

Outdent

Shifts all lines in the selection to the previous tab stop

Toggle Breakpoint

Sets or removes a breakpoint at the current line.

Comment Block

Adds comment character to the beginning of each line of a selected block of text.

Uncomment Block

Removes the comment character from each line of a selected block of text.

Toggle Bookmark

Toggles a bookmark on or off for the active line in the Code window.

Next Bookmark

Moves the focus to the next bookmark.

Previous Bookmark

Moves the focus to the previous bookmark.

Clear All Bookmarks

Removes all bookmarks set in a project.

Debug Toolbar

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdDebugCommandBarVB5C;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdDebugCommandBarVB5S"}

Contains buttons that are shortcuts to some commonly used menu items frequently used in debugging code.

You can click a toolbar button once to carry out the action represented by that button. You can select the Show ToolTips option in the General tab of the Options dialog box if you want to display ToolTips for the toolbar buttons.

Toolbar Buttons

Start

Runs the application from the startup form (or Sub Main) specified in the Project tab of the Options dialog box. The Command button changes to Continue if in break mode.

Break

Stops execution of a program temporarily. Click the Continue button to resume running a program.

End

Stops running the program and returns to design time.

Toggle Breakpoint

Sets or removes a breakpoint at the current line

Step Into

Executes code one statement at a time.

Step Over

Executes code as a unit when the current statement contains a call to a procedure, and then steps to the next statement in the current procedure.

Step Out

Executes the remaining lines of a function in which the current execution point lies.

Locals Window

Displays the Locals window.

Immediate Window

Displays the Immediate window.

Watch Window

Displays the Watch window.

Quick Watch

Displays the Quick Watch dialog box with the current value of the selected expression.

Call Stack

Displays the Calls dialog box, which lists the currently active procedure calls–procedures in the application that have started but are not completed.

Form Editor Toolbar

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdDialogCommandBarC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdDialogCommandBarS"}

Contains buttons that are shortcuts to some commonly used menu items useful for working with forms.

You can click a toolbar button once to carry out the action represented by that button. You can select the Show ToolTips option in the General tab of the Options dialog box if you want to display ToolTips for the toolbar buttons.

Toolbar Buttons

Bring To Front

Moves the selected objects to the front of all other objects on a form.

Send To Back

Moves the selected objects behind all other objects on a form.

Align

Lefts — Aligns the horizontal position of the selected objects, putting the left-most edges in line with the last selected object, the one with the black grab handles.

Centers — Aligns the horizontal position of the selected objects, putting the centers in line with the last selected object, the one with the black grab handles.

Rights — Aligns the horizontal position of the selected objects, putting the right-most edges in line with the last selected object, the one with the black grab handles.

Tops — Aligns the vertical position of the selected objects, putting the tops in line with the last selected object, the one with the black grab handles.

Middles — Aligns the vertical position of selected objects, putting the middles in line with the last selected object, the one with the black grab handles.

Bottoms — Aligns the vertical position of the selected objects, putting the bottoms in line with the last selected object, the one with the black grab handles.

to Grid — Aligns the top left of the selected objects to the closest grid. The object is not resized.

Center

Horizontally — Aligns the middles of the selected objects to a horizontal line in the middle of the form.

Vertically — Aligns the middles of the selected objects to a vertical line in the center of the form.

Make Same Size

Width — Adjusts width.

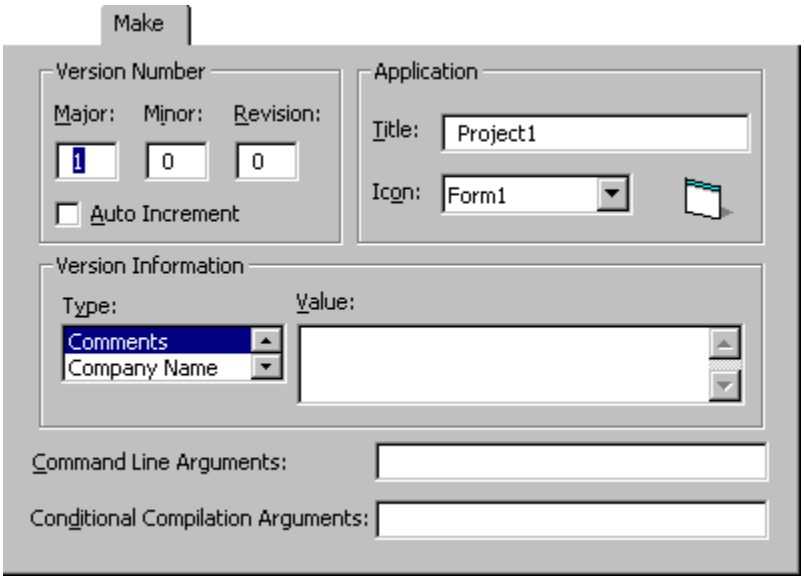
- Height — Adjusts height.
- Both — Adjusts both the width and the height.

Lock Controls Toggle

Locks and unlocks controls. When controls are locked, all controls on the form are kept in their current positions so you don't inadvertently move them once they are in the desired location.

Make Tab (Project Properties Dialog Box)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgBuildOptionsTabProjectSettingsDialogC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgBuildOptionsTabProjectSettingsDialogS"}



Sets the attributes for the executable file you make. Displays the name of the current project in the title so you can determine which project will be affected by any changes you make. The current project is the item currently selected in the Project Explorer.

Tab Options

Version Number Creates the version number for the project.

- Major — Major release number of the project; 0 – 9999.
- Minor — Minor release number of the project; 0 – 9999.
- Revision — Revision version number of the project; 0–9999.
- Auto Increment — If selected, automatically increases the Revision number by one each time you run the Make Project command for this project.

Application Lets you identify a name and icon for your project.

- Title — Name of the application.
- Icon — Icon for the application.

Version Information Lets you provide specific information about the current version of your project.

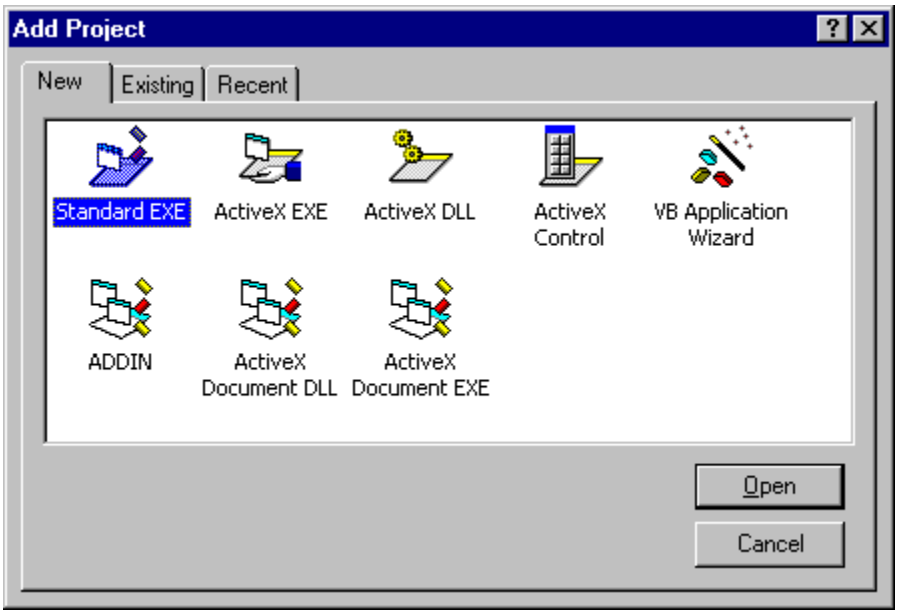
- Type — Information you can use to set a value. You can enter information for your company name, file description, legal copyright, legal trademarks, product name and comments.
- Value — The value for the type of information selected in the Type box.

Command Line Arguments Allows you to enter the command-line arguments that Visual Basic sends to an application only when you choose Start from the Run menu.

Conditional Compilation Arguments Allows you to enter the constant declarations used for conditional compilation.

Add Project Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgAddProjectC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgAddProjectS"}



Displays the project types or existing projects you can add to your project. When you add a project, a project group is created, if one does not already exist. A project group is useful if you want to load a set of related projects at the same time.

Dialog Box Options

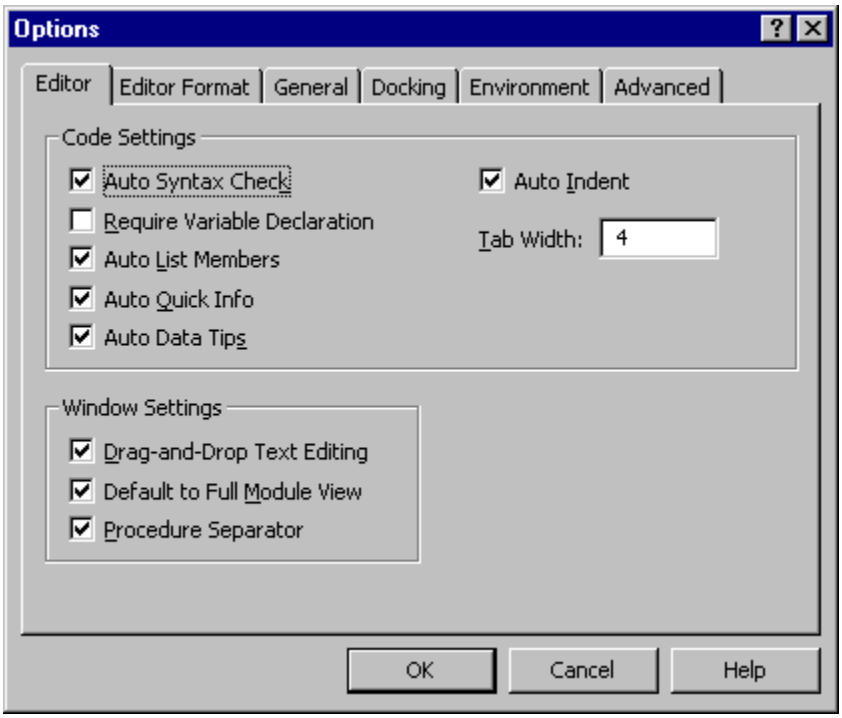
New Tab Displays the project types you can add.

Existing Tab Displays a dialog box where you can locate and select the project that you want to add.

Recent Tab Lists the most recently opened projects and their location.

Options Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgOptionsDialogVB5C;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgOptionsDialogVB5S"}



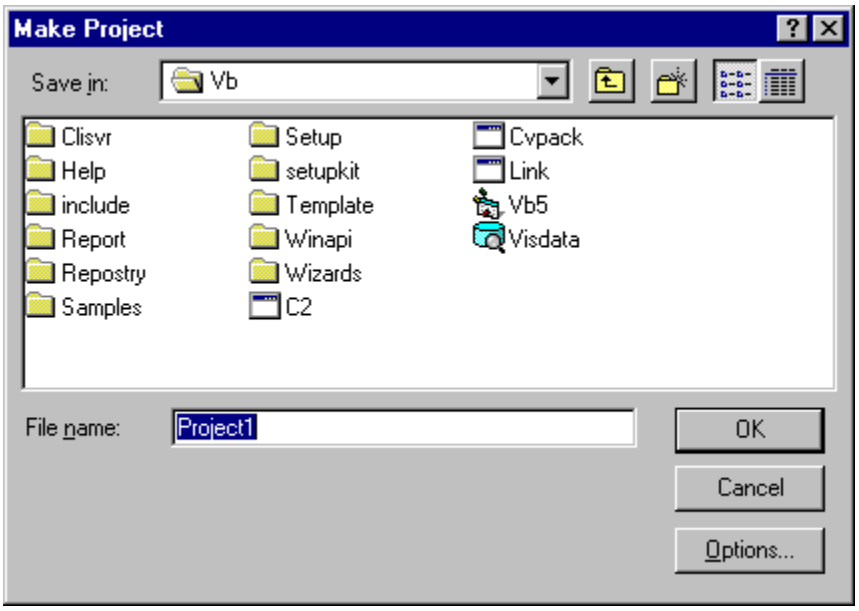
Allows you to change default settings for Visual Basic development environment.

The Options dialog box has the following tabs:

- Editor
- Editor Format
- General
- Docking
- Environment
- Advanced

Make Project Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgMakeProjectDialogC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgMakeProjectDialogS"}



Allows you to build any combination of projects in the project group.

Dialog Box Options

Save in

Select the folder where you want to store the file.

Up One Level

Shows a list of folders or drives one level above the current folder.

Create New Folder

Creates a new folder.

Details

Shows the folder or documents in a list that includes the folder or document icon and name, its size (documents only), type, and the date and time it was last modified.

List

Shows the folders or documents in a list format that includes the folder or document icon and its name.

File name

Allows you type a name for your project. Visual Basic automatically adds the default filename extension .vbg.

OK

Saves the project under the name and in the location specified.

Cancel

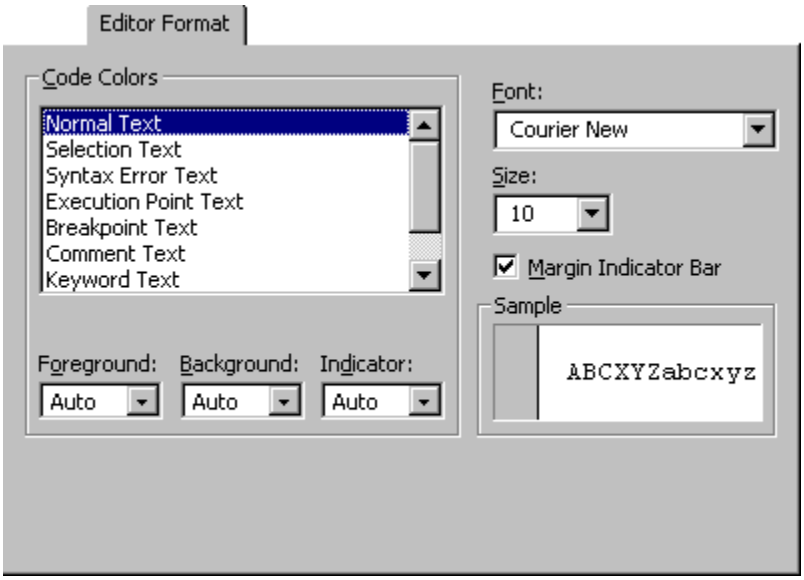
Closes the dialog box without saving the file.

Options

Displays the Project Properties dialog box with the Make tab active. You can indicate the version number, information, and compatibility and the icon for your project.

Editor Format Tab (Options Dialog Box)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnEditorFormatTabOptionsDialogBoxC"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnEditorFormatTabOptionsDialogBoxS"}



Specifies the appearance of your Visual Basic code.

Tab Options

Code Colors Determines the foreground and background colors used for the type of text selected in the list box.

- Text List — Lists the text items that have customizable colors.
- Foreground — Specifies the foreground color for the text selected in the Color Text List.
- Background — Specifies the background color for text selected in the Color Text List
- Indicator — Specifies the margin indicator color.

Font Specifies the font used for all code.

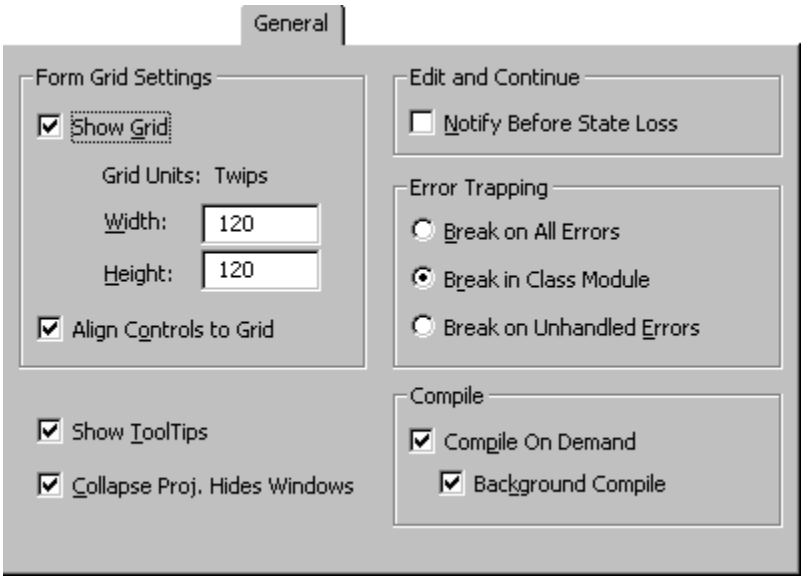
Size Specifies the size of the font used for code.

Margin Indicator Bar Makes the margin indicator bar visible or not visible.

Sample Displays sample text for the font, size, and color settings.

General Tab (Options Dialog Box)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgOptionsGeneralTabC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgOptionsGeneralTabS"}



Specifies the settings, the error handling, and compile settings for your current Visual Basic project.

Tab Options

Form Grid Settings Determines the appearance of the form grid at design time.

- Show Grid — Determines whether to show the grid at design time.
- Grid Units — Displays the grid units used for the form. The default is twips.
- Width — Determines the width of grid cells on a form; 24–1,188 twips.
- Height — Determines the height of grid cells on a form; 24–1,188 twips.
- Align Controls to Grid — Automatically positions the outer edges of controls on grid lines.

Show ToolTips Displays ToolTips for the toolbar and Toolbox items.

Collapse Proj. Hides Windows Determines whether the window are hidden when a project is collapsed in the Project Explorer.

Edit and Continue

- Notify Before State Loss — Determines if you will receive a message notifying you that the action requested will cause the state to be lost for a running project.

Error Trapping Determines how errors are handled in the Visual Basic development environment. These settings are not saved for each project so setting this option affects all instances of Visual Basic started after you change the setting.

- Break on All Errors — Any error causes the project to enter break mode, whether or not an error handler is active and whether or not the code is in a class module.
- Break in Class Module — Any unhandled error produced in a class module causes the project to enter break mode at the line of code in the class module which produced the error.

  When you debug an ActiveX server project by running an ActiveX client test program in another project, set this option in the ActiveX server project to break on errors in its class modules, instead of always returning the error to the client test program.
- Break on Unhandled Errors — If an error handler is active, the error is trapped without entering

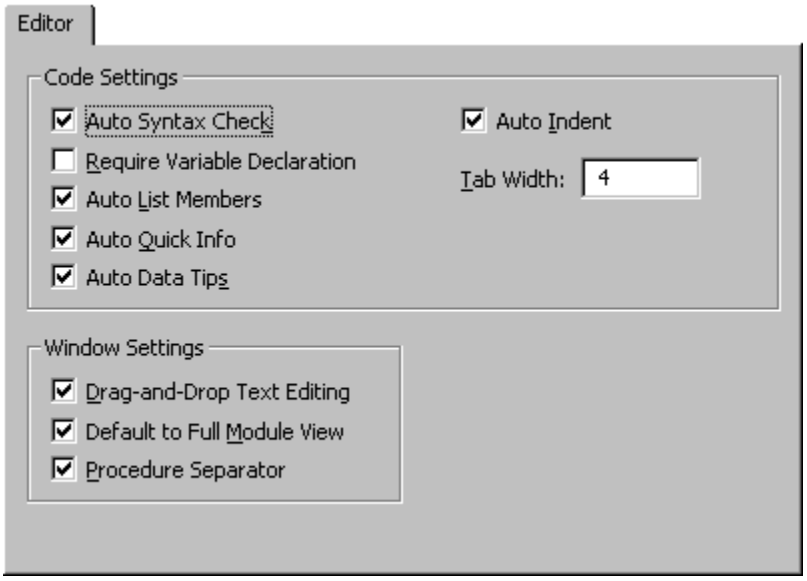
break mode. If there is no active error handler, the error causes the project to enter break mode. An unhandled error in a class module, however, causes the project to enter break mode on the line of code that invoked the offending procedure of the class.

Compile Determines how your project compiles.

- Compile On Demand — Determines whether a project is fully compiled before it starts, or whether code is compiled as needed, allowing the application to start sooner. If you choose the Start With Full Compile command on the Run menu, Visual Basic ignores the Compile on Demand setting and performs a full compile.
- Background Compile — Determines whether idle time is used during run time to finish compiling the project in the background. Background Compile can improve run time execution speed. This feature is not available unless Compile on Demand is also selected.

Editor Tab (Options Dialog Box)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbrgnFormatOptionsC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbrgnFormatOptionsS"}



Specifies the code window and project window settings.

Tab Options

Code Settings

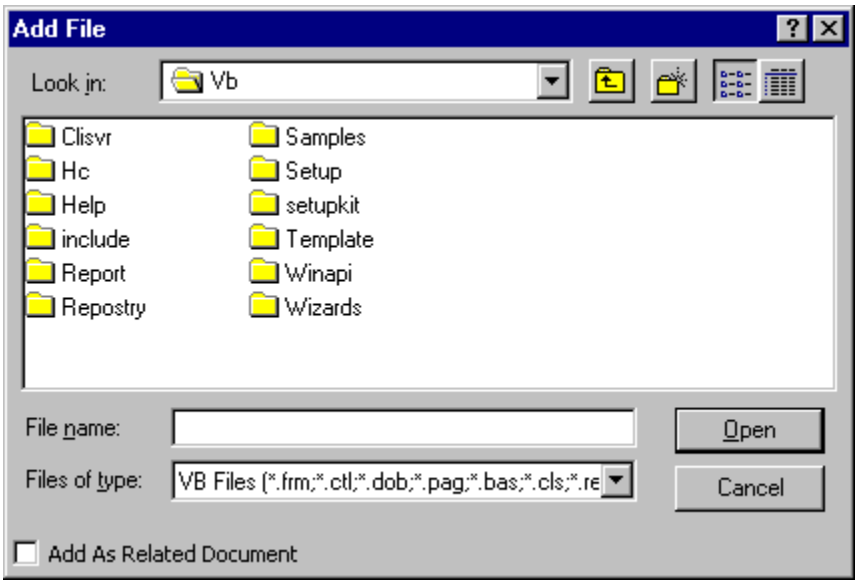
- Auto Syntax Check — Determines whether Visual Basic should automatically verify correct syntax after you enter a line of code.
- Require Variable Declaration — Determines whether explicit variable declarations are required in modules. Selecting this adds the Option Explicit statement to general declarations in any new module.
- Auto List Members — Displays a box that displays information that would logically complete the statement at the current insertion point.
- Auto Quick Info — Displays information about functions and their parameters.
- Auto Data Tips — Displays the value of the variable over which your cursor is placed
- Auto Indent — Allows you to tab the first line of code; all subsequent lines will start at that tab location.
- Tab Width — Sets the tab width, which can range from 1 to 32 spaces; the default is 4 spaces.

Window Settings

- Drag-and-Drop Text Editing — Allows you to drag and drop elements within the current code and from the Code window into the Immediate or Watch windows,
- Default to Full Module View — Sets the default state for new modules to allow you to look at procedures in the Code window either as a single scrollable list or one procedure at a time. It does not change the way currently open modules are viewed.
- Procedure Separator — Allows you to display or hide separator bars that appear at the end of each procedure in the Code window. Only available if Default to Full Module View is checked.

Add File Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgAddFileDialogC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgAddFileDialogS"}



Allows you to insert a file into your project.

Dialog Box Options

Look in

Select the location of the project you want to open.

Up One Level

Shows a list of folders or drives one level above the current folder.

Create New Folder

Creates a new folder.

List

Shows the folders or documents in a list format that includes the folder or document icon and its name.

Details

Shows the folder or documents in a list that includes the folder or document icon and name, its size (documents only), type, and the date and time it was last modified.

File name

Select or type the name of a project you want to open.

Files of type

Select a file type; the default is Text Files (*.txt). Files of the selected type will appear in the File Name list box.

Add As Related Document

Adds items such as Microsoft Excel or Microsoft Word, to the Related Documents folder in the Project Explorer.

Open

Inserts the selected file.

Cancel

Closes the dialog box without inserting a new project.

Compile Tab (Project Properties Dialog Box)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgCompileTabProjectPropertiesC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgCompileTabProjectPropertiesS"}

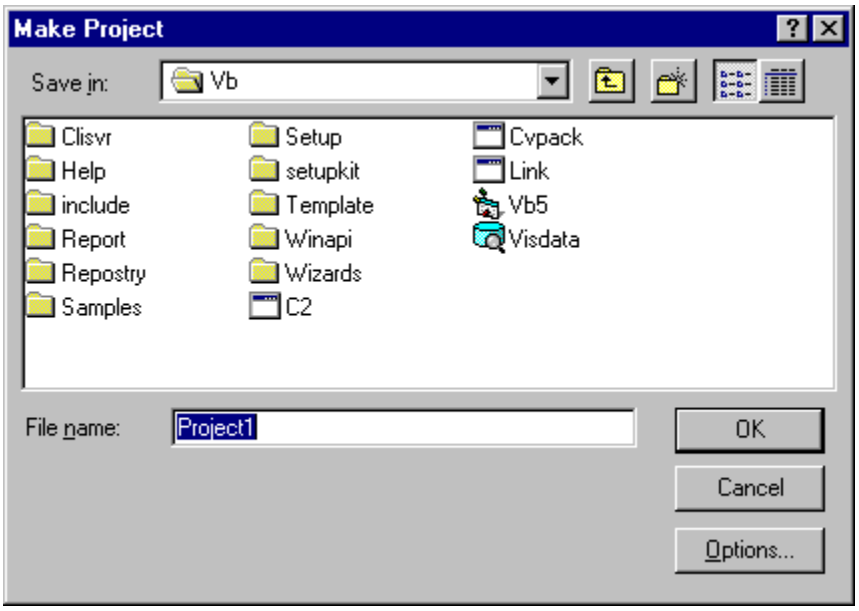
Allows you to set conditions for compiling your project.

Tab Options

DLL Base Address Sets a base address for the program, overriding the default location for a .dll file (at 0X10000000). The operating system first attempts to load a program at its specified or default address. If there is insufficient space, the system relocates the program.

Make Project Group Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgMakeProjectGroupDialogBoxC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgMakeProjectGroupDialogBoxS"}



Allows you to choose the available projects that you want to build.

Dialog Box Options

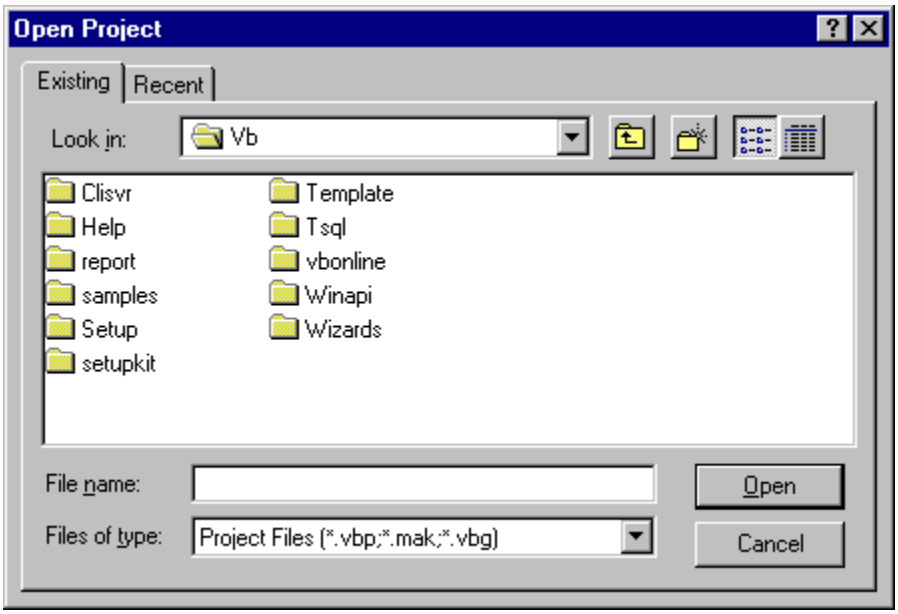
Project List Lists the projects that are available. You can choose as many projects as you want to build together

Use Default Build Options When checked, uses the options set in the Make tab of the Project Properties dialog box. If you have not changed the options, Visual Basic will use the default values and the project name for the build file names.

Build Builds the project files.

Open Project Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgOpenProjectDialogC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgOpenProjectDialogS"}



Lets you locate and open a project or project group.

Dialog Box Options

Existing Tab Displays a dialog box where you can locate and select the project that you want to open.

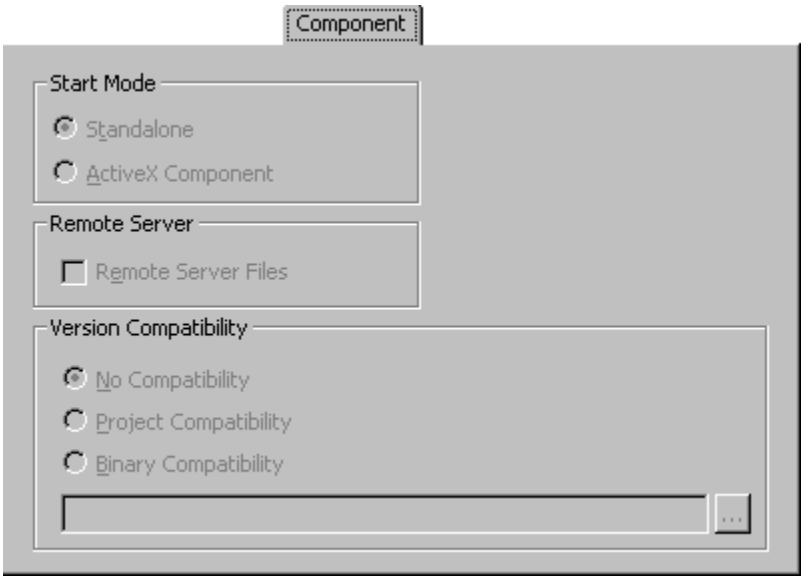
Recent Tab Lists the most recently opened projects and their location.

Open Displays the selected file.

Cancel Closes the dialog box without opening a new project.

Component Tab (Project Properties Dialog Box)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgStartUpTabProjectPropertiesC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgStartUpTabProjectPropertiesS"}



Allows you to determine how your project starts.

Tab Options

Start Mode Indicates how the application is started. This is used with ActiveX servers (Visual Basic, Professional and Enterprise Editions).

- Standalone — Starts the application a standalone.
- ActiveX Component — Starts the application as an ActiveX Server.

Remote Server

- Remote Server Files (Enterprise Edition) — When checked, Visual Basic creates a file with a .vbr file name extension and the same filename as the .dll file. This .vbr file contains information needed by the Windows Registry to run an ActiveX Server on a remote computer. This option is only available in the Enterprise edition of Visual Basic.

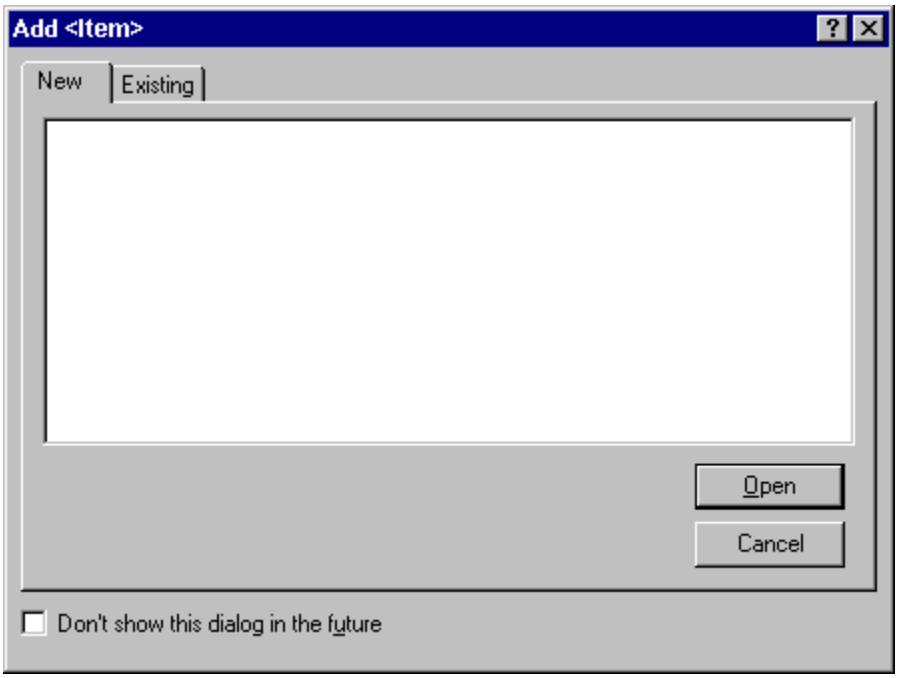
Version Compatibility Allows you to set the level of version compatibility

- No Compatibility — Compatibility not enforced.
- Project Compatibility — If checked, the Location box active and allows you to search for the file with which this project is to be compatible. If cleared, the Location box is not available.

  For all ActiveX project types, Project Compatibility is checked by default.
- Binary Compatibility — Useful for maintaining compatibility among projects that have been compiled using your component.
- File Location Box — Displays the name and location of the file with the project is to be compatible. You can type a name and location or can use the Browse button to display the Compatible ActiveX Server dialog box where you can locate the file.

Add <item> Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgAddItemDialogC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgAddItemDialogS"}



Allows you to locate and add a form, MDI form, module, class module, user control, property page, user document, or ActiveX designer to your project.

Dialog Box Tabs

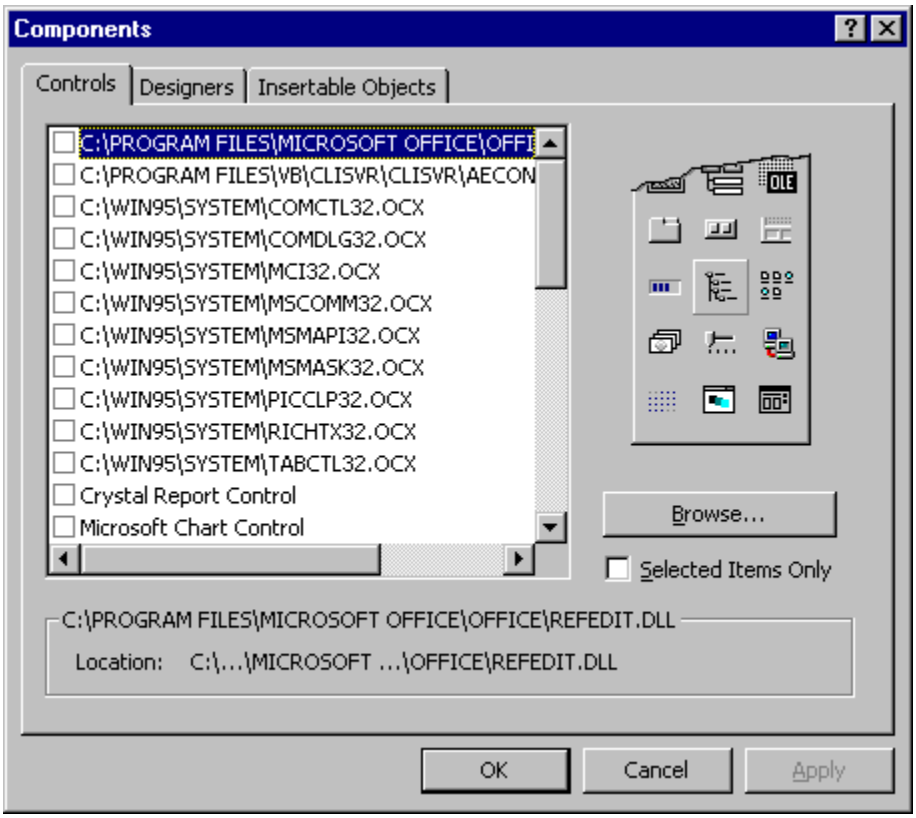
New Lists the available items.

Existing Displays the common Open dialog box where you can locate and open the item you want to add.

Don't show this dialog in the future When selected automatically adds an new item to your project without showing the Add <item> dialog box. To view the Add <item> dialog box in the future select the Prompt for project option on the Environment tab of the Options dialog box.

Components Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgComponentsDialogC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgComponentsDialogS"}



Allows you to add components – controls, designers, or insertable objects – to your project.

You can reference loaded control projects from the Components dialog box.

Note Pressing SHIFT while clicking OK removes unused control references from your project.

Dialog Box Options

Tabs List components you can add by their category.

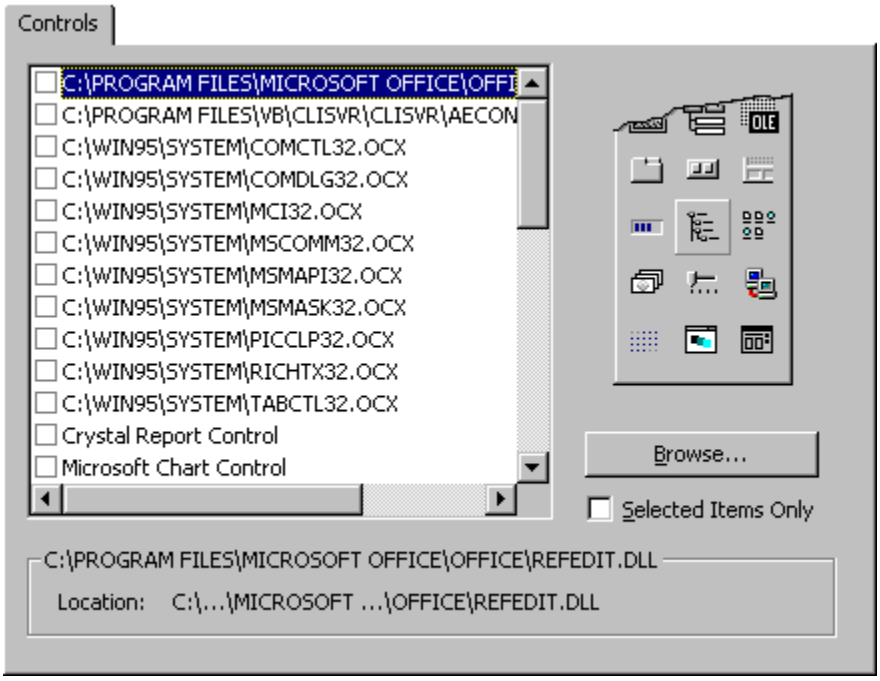
- Controls
- Designers
- Insertable Objects

Apply Adds the selected controls, designers, or insertable objects to the Toolbox without closing the dialog box.

OK Adds the selected controls, designers, or insertable objects to the Toolbox and closes the Components dialog box.

Controls Tab (Components Dialog Box)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgControlsTabComponentsDialogBoxC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgControlsTabComponentsDialogBoxS"}



Allows you to add controls to your project's Toolbox.

Tab Options

Available Controls List Displays the available controls. You add a control by selecting the check box next to its name.

Note You can't remove a control that is used in your project.

Path Displays the path of the control selected in the Available Controls box.

Browse Displays the Add Custom Control dialog box where you can locate and open the control you want.

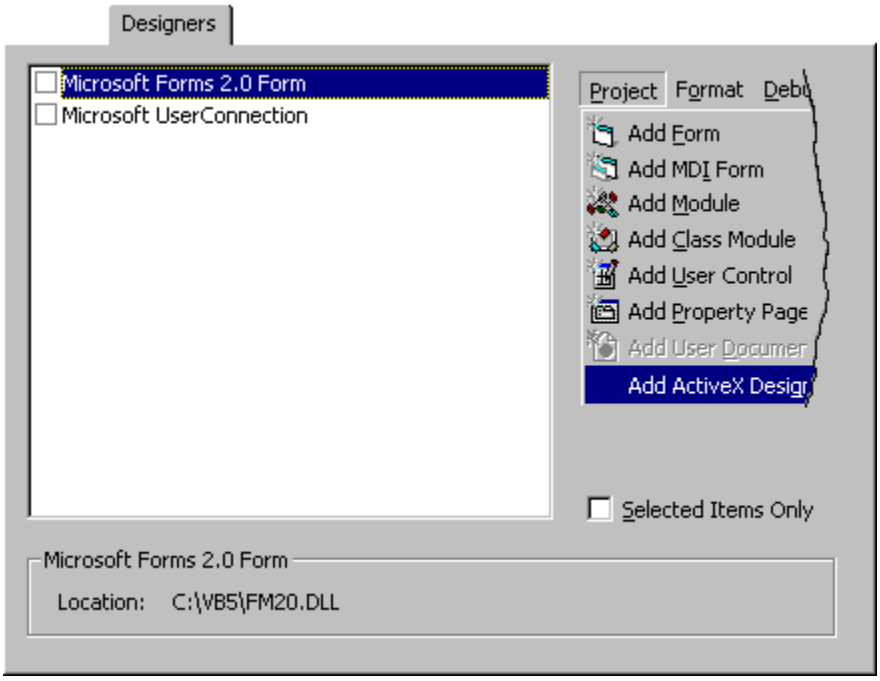
Selected Items Only When selected, displays only those controls in the Available Controls list which you have selected to include in the project.

Apply Adds the selected controls to the Toolbox without closing the dialog box.

OK Adds the selected controls to the Toolbox and closes the Components dialog box.

Designers Tab (Components Dialog Box)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgDesignersTabComponentsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgDesignersTabComponentsS"}



Allows you to add designers to your project's Toolbox.

Tab Options

Available Designers List Displays the available designers. You add a designer by selecting the check box next to its name.

Note You can't remove a designer that is used in your project.

Path Displays the path of the designer selected in the Available Designers List.

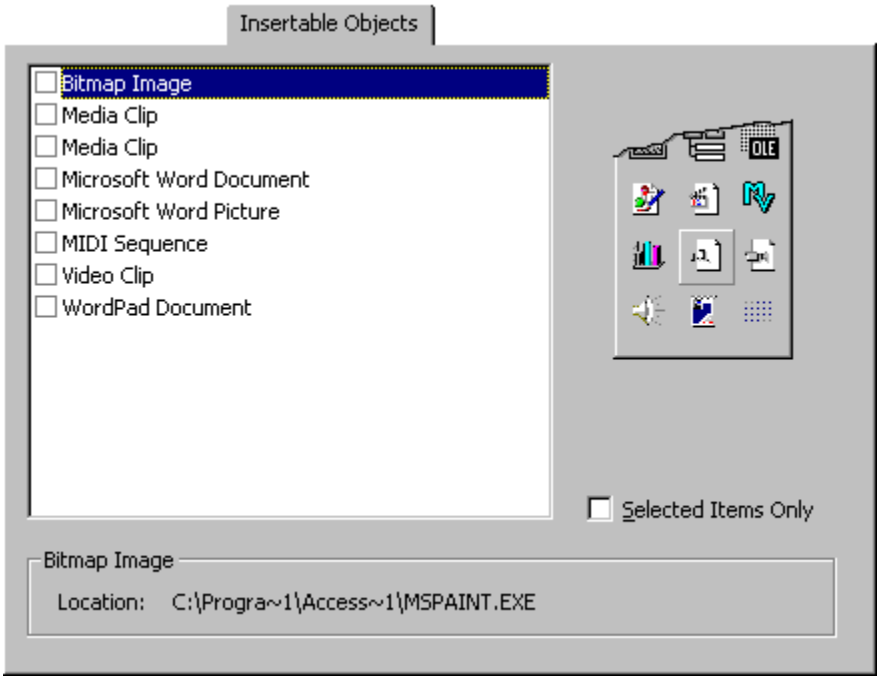
Selected Items Only When selected, displays only those designers in the Available Designers list which you have selected to include in the project.

Apply Adds the selected designers to the Toolbox without closing the dialog box.

OK Adds the selected designers to the Toolbox and closes the Components dialog box.

Insertable Objects Tab (Components Dialog Box)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgInsertableObjectsTabComponentsC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgInsertableObjectsTabComponentsS"}



Allows you to add insertable objects to your project's Toolbox.

Tab Options

Available Insertable Objects Displays the available insertable objects. You add an insertable object by selecting the check box next to its name.

Note You can't remove an insertable object that is used in your project.

Path Displays the path of the insertable object selected in the Available Insertable Objects box.

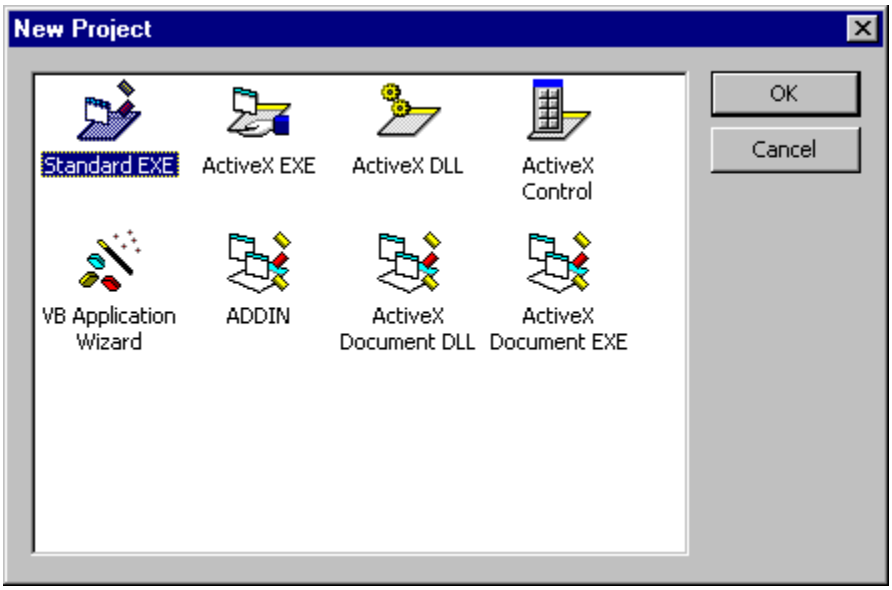
Selected Items Only When selected, displays only those insertable objects in the Available Insertable Objects list which you have selected to include in the project.

Apply Adds the selected insertable objects to the Toolbox without closing the dialog box.

OK Adds the selected insertable objects to the Toolbox and closes the Components dialog box.

New Project Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgNewProjectDialogOpeningC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgNewProjectDialogOpeningS"}



Allows you to select the type of project you want to create. If there is currently another project or project group open, you will be asked to save any changes before the new project and group are created.

If you want to add projects to a project group use the Add Project command on the File menu.

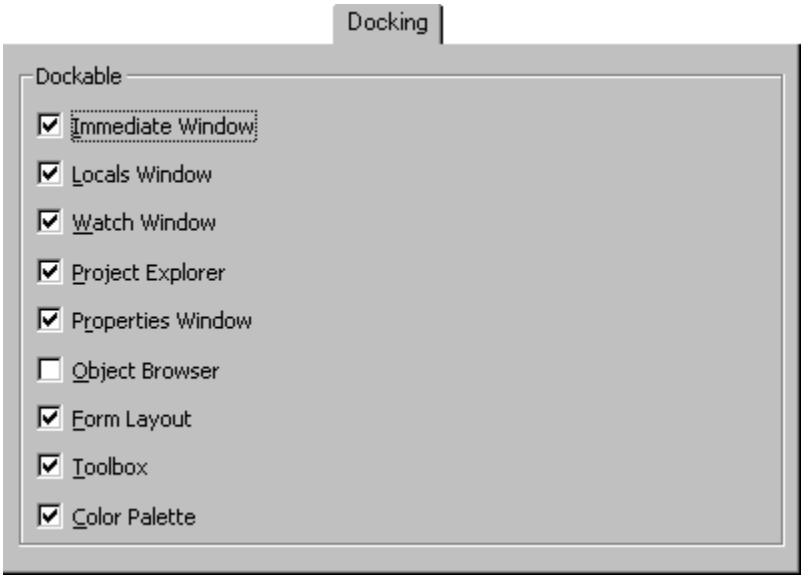
Dialog Box Options

List of projects Displays the valid projects that you can add.

- Standard EXE — Creates a standard executable file.
- ActiveX EXE — Creates an ActiveX executable file.
- ActiveX DLL — Creates an ActiveX DLL file.
- ActiveX Control — Creates an ActiveX control.
- Any projects with a .vbp (project files), .vbz (wizard files), or .vbg (project group) extension that are in the vb\template\projects directory.

Docking Tab (Options Dialog Box)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgDockingTabOptionsDialogC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgDockingTabOptionsDialogS"}



Allows you to choose which windows you want to be dockable.

A window is docked when it is attached or "anchored" to other windows that are dockable or to the main window when you are in MDI mode. When you move a dockable window, it "snaps" to the location. A window is not dockable when you can move it anywhere on the screen and leave it there.

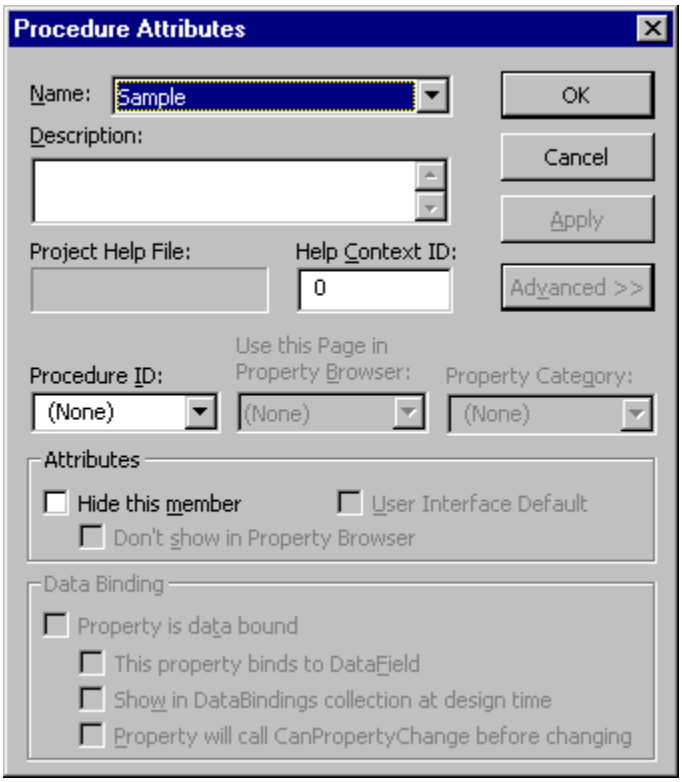
Tab Options

Dockable List the windows that are dockable.

Select the windows you want to be dockable and clear those that you do not. You can have any, none, or all of the windows in the list dockable.

Procedure Attributes Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgProcedureAttributesDialogC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgProcedureAttributesDialogS"}



Allows you to set the attributes for properties, methods, and events specified for a an item.

Only available if you have one or more procedures defined.

Dialog Box Options

Name Lists the properties, methods, and events defined. The property, method, or event in the Code window in which the cursor is currently positioned appears selected.

Description Displays the description of the property, method, or event that you want to show up in the Object Browser.

Project Help File Displays the path to the help file for the project that is specified on the General tab of the Project Properties dialog box. This option is read only.

Help Context ID Specifies the help context ID specified on the General tab of the Project Properties dialog box for the selected property or method.

OK Applies the options to the selected property or method on the user control and closes the dialog box.

Apply Applies the options to the selected property or method on the user control without closing the dialog box.

Advanced Expands the dialog box to include the following options:

Procedure ID Allows you to choose a standard member ID for the selected property, event, or method. The member ID is used to identify a property, method, or event as a standard type that control hosts may know about. A procedure whose ProcedureID is 0 acts as the default property or method for the control. It is also referred to as the **Value** property.

Note When you set a property, method, or event as a standard type, it does not change the behavior of the control. It tells a control container that the property, method, or event will behave in the well-understood way but it does not create the behavior. It is up to you to make sure that the control behaves correctly.

Use this Page in Property Browser Lists the Property Pages that are in the current project so you can select one to act as a builder when the property is chosen in the Property window. The property in the Property window is marked with a button and an ellipsis (…). When a user clicks the ellipsis, the Property Page specified is displayed.

Valid only for properties. Default is None.

Property Category Lists available categories to describe the selected property. You can select a standard category or type in one of your own. When you type in a category, the property browser automatically creates a new level for the property.

Some property browsers such as Visual Basic allow you to categorize control properties. If the control host does not support property categorization, this setting will be ignored.

Valid only for properties. Default is None.

Attributes Allows you to set some standard behaviors of the selected property, method, or event.

- Hide this member — Determines whether the property, method, or event will appear to the end user of the control. If checked, it is hidden and cannot be seen in a property browser or the Object Browser. You can still write code to access it but it does not appear in the user interface.
- User Interface Default — Determines which property is highlighted in the property browser or which event is displayed in the Code window when you double-click the control. There can be only one User Interface default property and one User Interface default event. Not valid for methods.
- Don't show in Property Browser — Determines if a property, method, or event will be hidden in the property browser. It will continue to appear in the Object Browser and you can continue to write code to access it. This box is cleared when you choose Hide this member.

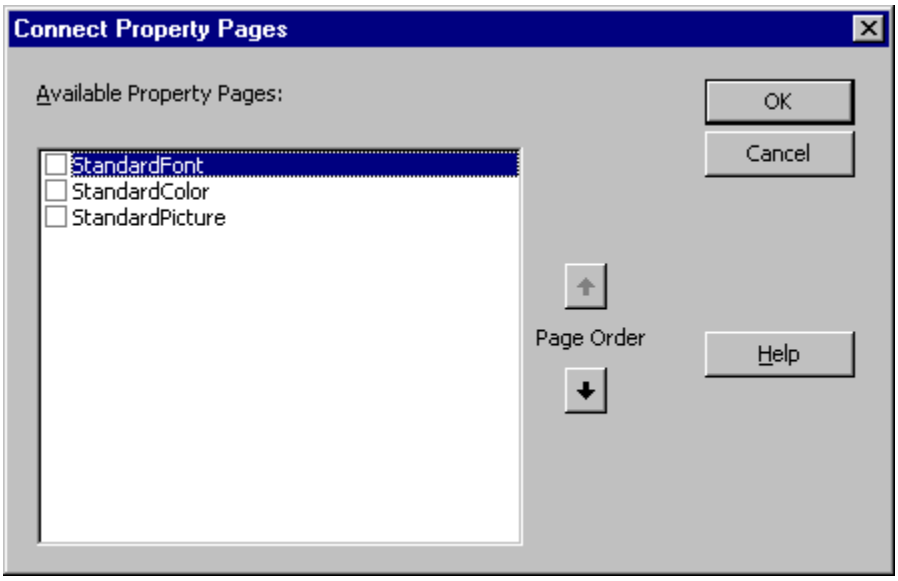
Data Binding Determines whether a property can be bound or linked to a field in a database table.

- Property is data bound — Determines if the property is a data bound property. If this option is selected, the property supports data binding and appears in the **DataBindings** collection. The box is cleared by default.
- This property binds to DataField — Specifies whether the field to which the property is bound is specified in the **DataField** property. With this option, the end user does not have to use the **Data Bindings** collection and can use **DataField**. The box is cleared by default.
- Show in DataBindings collection at design time — Determines if the property will appear in the user interface as bindable at design time. If selected, the property appears in the Bindings dialog box. If cleared, the property does not appear in the user interface as a bindable property but you can continue to write code to access it.
- Property will call **CanPropertyChange** before changing — Tells the control container that the control will always call the **CanPropertyChange** method, and respect the return value, before changing a property value.

  The box is cleared by default.

Connect Property Pages Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgConnectPropertyPagesDialogC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgConnectPropertyPagesDialogS"}



Allows you to set the property pages at design time.

Dialog Box Options

Available Property Pages Lists the available property pages. You check the pages that you want to appear on the control.

If you delete a property page from the project that was selected to be used with a control, it appears with the word, MISSING, as a prefix. The next time you open this box, the deleted property page does not appear in the list.

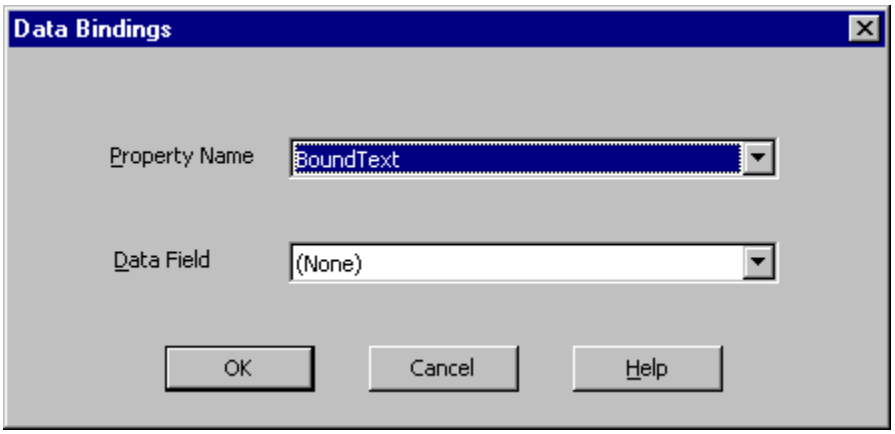
Page Order Sets the order that the pages appear in the Properties window. The page at the top of the list is the first page that appears when you open the Properties window.

Moves the selected Property Page up one level each time you click it.

Moves the selected Property Page down one level each time you click it.

Data Bindings Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdBindingdDialogBoxC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdBindingdDialogBoxS"}



Allows you to set the Data Field for each property that is marked as **Bindable** or **DisplayBind**.

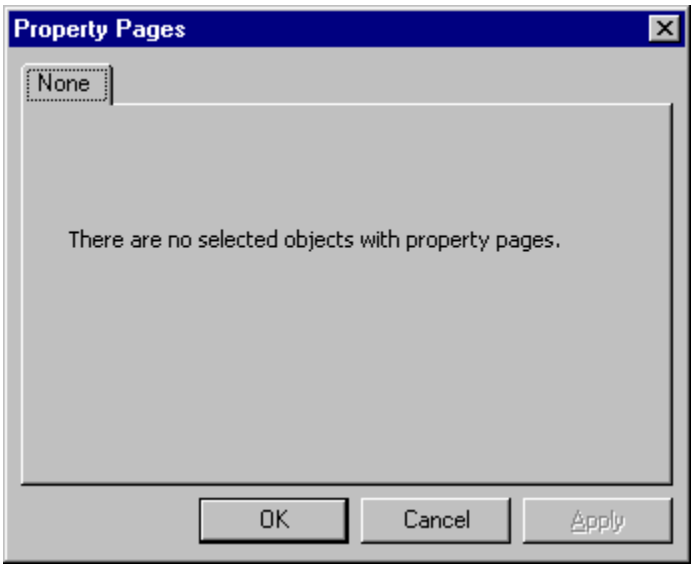
Dialog Box Options

Property Name Displays a list of the properties that are marked as **Bindable** and **DisplayBind**. A property that has a data field assigned to it appears in Bold text.

Data Field Lists all of the fields from the source specified in the **DataSource** property of the control.

Property Pages Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgPropertyPagesDialogC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgPropertyPagesDialogS"}



Allows you to change a control's properties at design time.

Dialog Box Options

Tabs Visual Basic creates a tabbed dialog box that acts like form by writing code to handle updating property values when a user changes values in the control.

You can add Property Pages to your project using the Add Property Pages command on the Project menu.

OK Adds the Property Pages and closes the Property Pages dialog box.

Apply Adds the Property Page without closing the dialog box.

Define Color Dialog Box

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgDefineColorDialogC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgDefineColorDialogS"}

Allows you to create customized colors.

Dialog Box Options

Color Screen with Slider Allows you to choose your color by sliding the bar up and down, from white to black.

Color Preview Box Allows you to preview your color.

Hue Adjusts the hue from 0-239 with 0 representing black and 239 representing white.

Sat Adjusts the color saturation from 0-240 with 0 representing black and 240 representing black.

Lum Adjusts the luminescence from 0-240 with 0 representing black and 240 representing white.

Red Adjusts the amount of red in the color from 0-255.

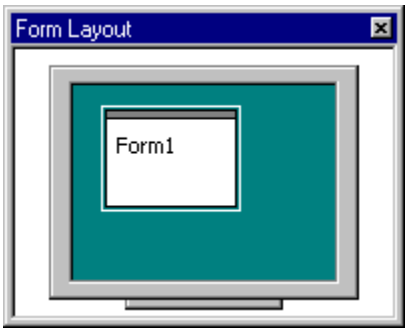
Green Adjusts the amount of green in the color from 0 to 255.

Blue Adjusts the amount of blue in the color from 0 to 255.

Add Color Adds your custom color to the Color Palette.

Form Layout Window

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



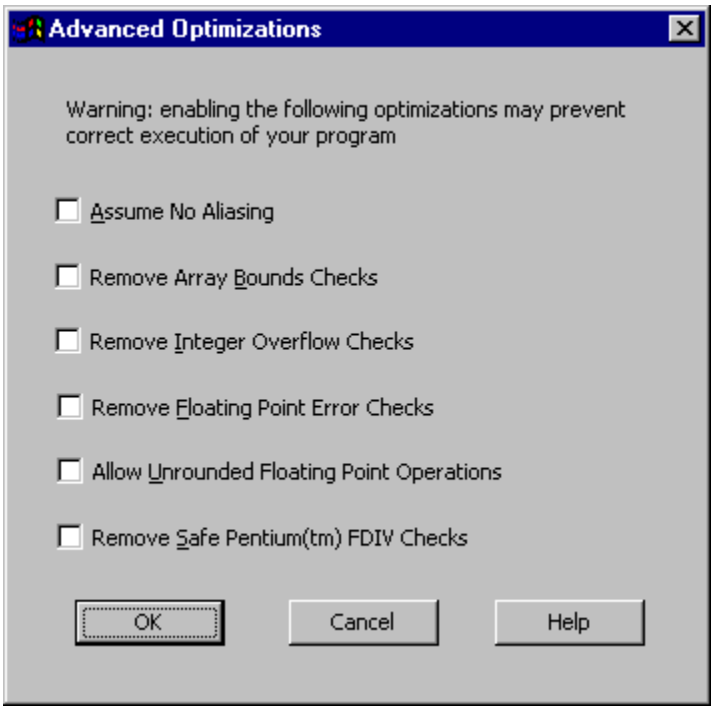
Allows you to visually position your forms at design time.

All forms that are visible in the environment are displayed. When you place your cursor over a form, it changes to a ⊞. If you press the mouse button you can position the form where you want it to appear at run time.

When you re-size the Form Layout window, each form is sized relative to the size of your design window. The upper left corner of the client area represents the coordinates – 0,0 – of the desktop. You can change the resolution using the Resolutions Guides command on the shortcut menu.

Advanced Optimizations Dialog Box

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



Allows you to add the following optimizations to your compile.

Note Enabling these optimizations may prevent the correct execution of your program.

Dialog Box Options

Assume No Aliasing Tells the compiler that your program does not use aliasing. Aliasing provides a name that refers to a memory location that is already referred to by a different name. Using this option allows the compiler to apply optimizations that it could not otherwise use, for example storing variables in registers and performing loop optimizations. This occurs when using ByRef arguments. For example,

```
Dim y as integer
Sub Foo(x as integer)
x=5   'Code is referring to the same variable (the
y=6   'global y)via two different names
End Sub
Sub Main
Foo y
End Sub
```

Remove Array Bound Checks By default in Visual Basic, a check is made on every access to an array to determine if the index is within the range of the array. If the index is not within array bounds an error message is displayed. Selecting this option urns off the array bounds error checking and removes checks for the correct number of dimension of the array.

Note This may speed up array manipulation but invalid memory locations may be accessed and result in unexpected behavior or program crashes.

Remove Integer Overflow Checks By default in Visual Basic, a check is made on every

calculation for integer-style data types – byte, integer, and long – to be sure that the value is within the range of the data type. If the magnitude of the value being put into the data type is incorrect an error occurs. Selecting this option turns off the error checking which can speed up integer calculations. However, if data type capacities are overflowed, no error occurs and you may get incorrect results.

Remove Floating Point Error Checks By default in Visual Basic, a check is made on every calculation of a floating point data type – Singe and Double – to be sure that the value is within range for that data type and that there are no divide by zero or invalid operations. If the magnitude of the value being put into the data type is incorrect, an error occurs. Selecting this option turns off the error checking which can speed up floating [point calculations. However, if data type capacities are overflowed, not error occurs and you may get incorrect result.

Allow Unrounded Floating Point Operations When selected, allows the compiler to:

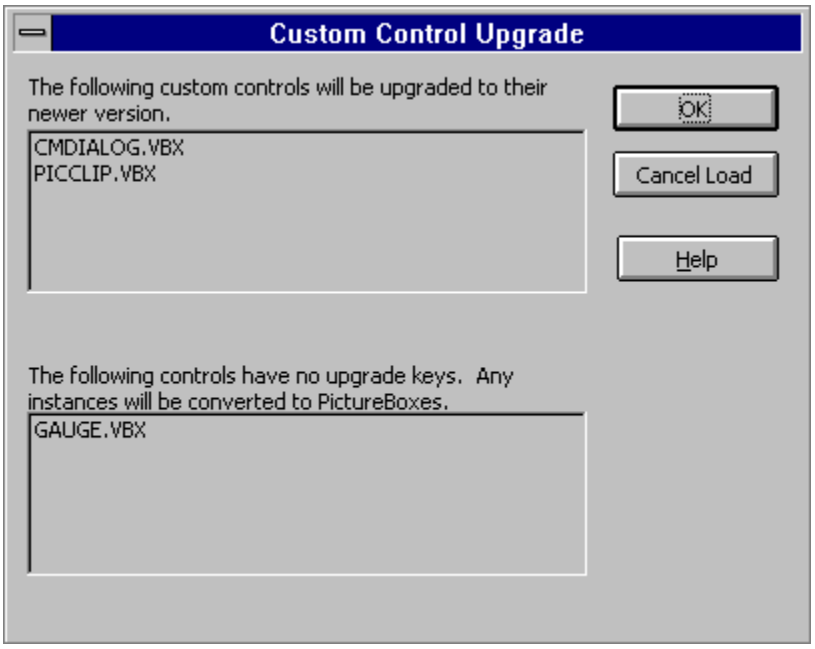
- Use floating point regisers more efficiently
- Avoid loads and stores from memory
- Do floating point comparisons more efficiently

Note Using this option may result in calculations being maintained to a higher precision than expected and may cause the comparison of two floating point values to show them unequal when you expect them to be equal.

Remove Safe Pentium™ FDIV Checks Removes the checking so that the code for floating point division is faster and smaller but may produce slightly incorrect results on Pentium™ processors with the FDIV bug. If cleared, the code generated for floating point division operations is not affected by the FDIV bug on Pentium™ processor.

Custom Control Upgrade Dialog Box

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



Allows Visual Basic to automatically upgrade old custom controls that you are using with appropriate ActiveX controls that are available on your system.

Dialog Box Options

Controls that can be upgraded Lists the controls that can be upgraded to their newer version.

Controls without upgrade keys Lists controls that have no upgrade keys and will be converted to PictureBoxes.

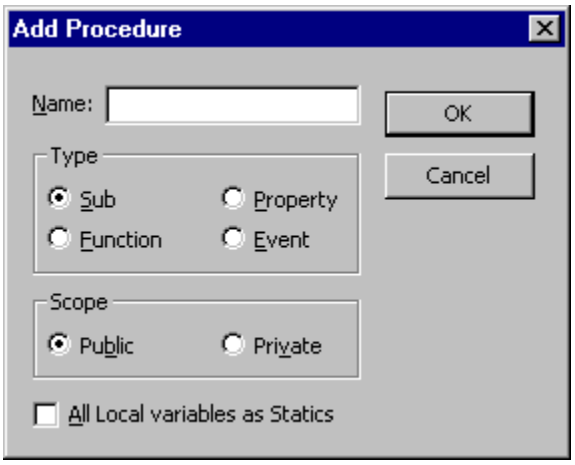
OK Replaces the controls with ActiveX controls.

Cancel Load Visual Basic does not replace the controls and cancels loading the project.

Note Replacement compatibility is determined by the ActiveX control vendor. Before replacing a custom control with an ActiveX control, be sure to check the ActiveX control vendor's documentation to see if you need to modify event procedures or other code associated with your project's use of the replacement controls.

Add Procedure Dialog Box

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



Inserts a new **Sub**, **Function**, or Property procedure. The Insert Procedure dialog box also allows you to set the scope to public or private, and make all local variables in the procedure static.

Dialog Box Options

Name Lists a name for the new procedure.

Type Identifies the type of procedure to create.

- Sub — Creates a new Sub procedure.
- Function — Creates a new Function procedure.
- Property — Creates a new Let property and Get property procedure pair.

Scope Sets the procedure's scope to either Public or Private.

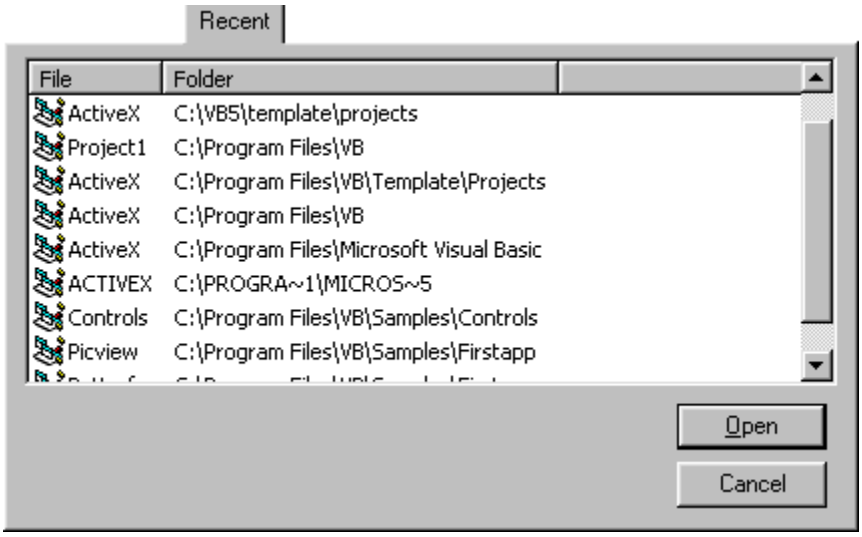
All Local Variables as Statics Adds the Static keyword to the procedure definition.

OK Inserts the procedure template in the Code window.

Cancel Closes the dialog box without inserting a new procedure.

Recent Tab (New, Open, and Add Project Dialog Boxes)

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



Displays a list and location of recently opened projects so you can open them or add them to your project.

Dialog Box Options

File Displays the list of recently opened project.

Folder Displays the path to the folder in which the file is located.

Open Opens the selected file and closes the dialog box.

New Project Dialog Box (at StartUp)

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

Allows you to locate and open an new or existing project when you open Visual Basic.

Dialog Box Tabs

New Tab Displays the project types you can open.

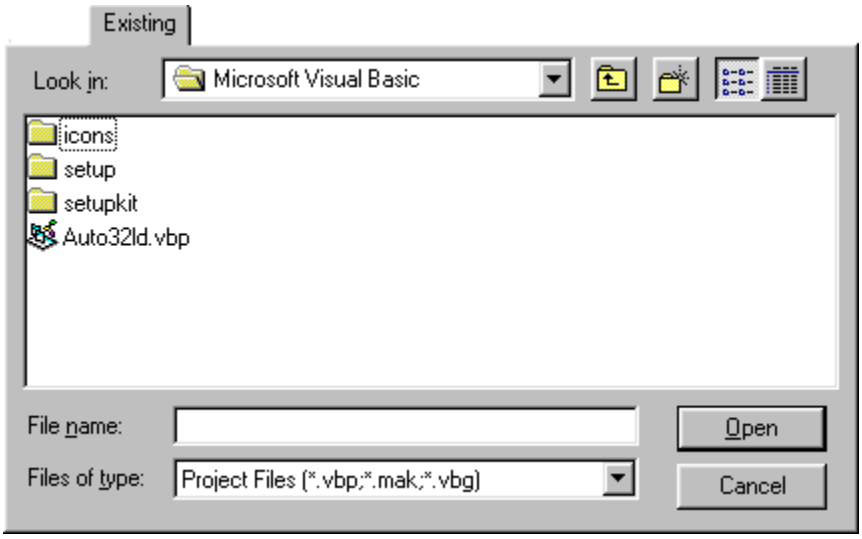
Existing Tab Displays a dialog box where you can locate and select the project that you want to open.

Recent Tab Lists the most recently opened projects and their location.

Don't show this dialog in the future When selected automatically opens a new Standard EXE project without showing the New Project dialog box. To view the New Project dialog box in the future select the Prompt for project option on the Environment tab of the Options dialog box.

Existing Tab (Add, New, and Open Projects Dialog Boxes)

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

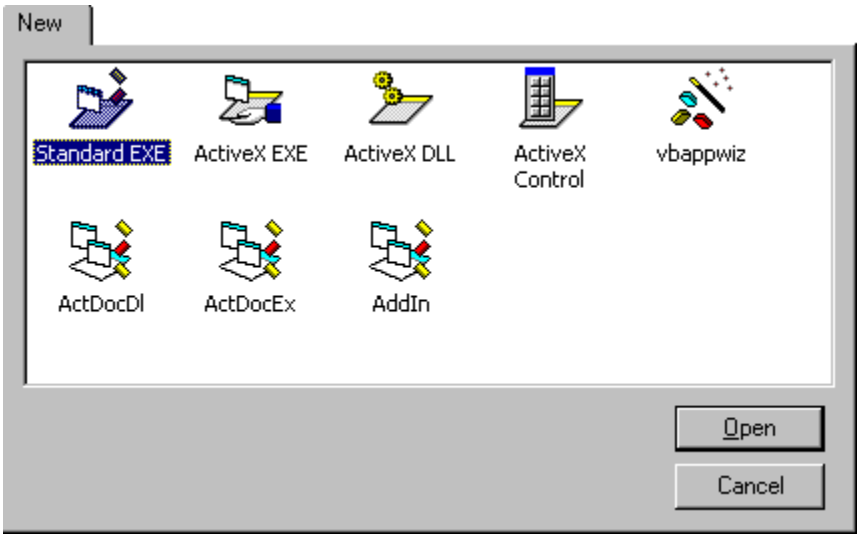


Displays a dialog box that looks like the Open common dialog box where you can locate and select the project that you want to add. When you add project groups, they merge into one project group.

You can add as many projects as your system resources and memory can accommodate.

New Tab (New Project and Add Project Dialog Boxes)

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



Note This tab is only available on the New Project dialog box that appears when you start Visual Basic and not on the New Project dialog box that you see when you choose New Project from the File menu.

Displays the project types you can add.

- Standard EXE — Creates a standard executable file.
- ActiveX EXE — Creates an ActiveX executable file.
- ActiveX DLL — Creates an ActiveX DLL file.
- ActiveX Control — Creates an ActiveX control.
- Any projects with a .vbp (project files) or .vbz (wizard files) extension that are in the vb\template\projects directory.

OLERequestPendingMsgText Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproOLERequestPendingMsgTextC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLERequestPendingMsgTextX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLERequestPendingMsgTextA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLERequestPendingMsgTextS"}

Returns or sets the text of the alternate "busy" message displayed when mouse or keyboard input is received while an automation request is pending. Not available at design time.

Syntax

object.**OLERequestPendingMsgText** [= *string*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	A string expression that evaluates to the message text that will be displayed in the alternate message box for the ActiveX request pending condition.

Remarks

Visual Basic displays a default Server Busy dialog box when mouse or keyboard input is received while an automation request is pending. This dialog box includes text and a Switch To button which are intended for use with visible ActiveX components such as Microsoft Excel. There are situations in which the default dialog box may not meet your needs:

- Your program may call a method of an object provided by an ActiveX component that has no user interface. ActiveX components created using Visual Basic Professional edition, for example, may run in the background without any visible forms.
- The ActiveX component you call may have been created using the Remote Automation features of Visual Basic, Enterprise edition, and may be running on another computer located at some distance from the user.
- If your program has loaded a Microsoft Excel workbook using the **GetObject** function, the workbook will not be visible when the user switches to Microsoft Excel. In fact, Microsoft Excel itself may not be visible, in which case the Switch To button does nothing.

In these situations, the default text and Switch To button are inappropriate and may confuse the user of your program.

The **OLERequestPendingMsgText** property allows you to replace the default Server Busy dialog box with an alternate message box. Setting **OLERequestPendingMsgText** to your own message string causes the default Server Busy dialog box to be replaced by a simple message box containing your message text and an OK button.

Note Once an automation request has been accepted by an ActiveX component, there is no way to cancel it.

If **OLERequestPendingMsgText** is equal to an empty string (""), the default Server Busy dialog is displayed.

Important When you know that an automation request may take more than a few seconds, and you are using a hidden or remote ActiveX component, you should set an alternate message. For remote ActiveX components, the alternate message is recommended for all requests. Network traffic may occasionally cause even a very short ActiveX request to take several seconds.

OLERequestPendingMsgTitle Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproOLERequestPendingMsgTitleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLERequestPendingMsgTitleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLERequestPendingMsgTitleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLERequestPendingMsgTitleS"}

Returns or sets the caption of the alternate "busy" message displayed when mouse or keyboard input is received while an automation request is pending. Not available at design time.

Syntax

object.**OLERequestPendingMsgTitle** [= *string*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	A string expression that evaluates to the caption of the alternate message box for the ActiveX request pending condition.

Remarks

If the **OLERequestPendingMsgText** property has been set, the value of the **OLERequestPendingMsgTitle** property is used as the caption of the alternate "busy" message box. The default value of the **OLERequestPendingMsgTitle** property is the current value of the **Title** property of the **App** object. This is the recommended setting.

If the **OLERequestPendingMsgText** property is set to an empty string (""), the **OLERequestPendingMsgTitle** property is ignored.

OLERequestPendingTimeout Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproOLERequestPendingTimeoutC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLERequestPendingTimeoutX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLERequestPendingTimeoutA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLERequestPendingTimeoutS"}

Returns or sets the number of milliseconds that must elapse before a "busy" message can be triggered by mouse or keyboard input received while an automation request is pending. Not available at design time.

Syntax

object.**OLERequestPendingTimeout** [= *milliseconds*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
milliseconds	A **Long** integer representing the number of milliseconds that must elapse before a busy message can be triggered.

Remarks

The default value of this property is 5000 milliseconds (five seconds).

Important This time-out value also affects documents you link or embed using the **OLE Container** control or the Toolbox. If you are using linked or embedded documents and you change this property before an automation request, it is a good idea to reset the value afterward.

OLEServerBusyMsgText Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproOLEServerBusyMsgTextC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEServerBusyMsgTextX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEServerBusyMsgTextA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEServerBusyMsgTextS"}

Returns or sets the text of the alternate "busy" message which is displayed if an ActiveX component rejects an automation request. Not available at design time.

Syntax

object.**OLEServerBusyMsgText** [= *string*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	A string expression that evaluates to the message text that will be displayed in the alternate message box for the ActiveX component busy condition.

Remarks

Visual Basic continues to retry an automation request for the number of milliseconds specified by the **OLEServerBusyTimeout** property. If the ActiveX component has not accepted the request in that interval, Visual Basic displays a default Server Busy dialog box. This dialog box includes text and a Switch To button which are intended for use with visible ActiveX components such as Microsoft Excel. There are situations in which the default dialog box may not meet your needs:

- Your program may call a method of an object provided by an ActiveX component that has no user interface. ActiveX components created using Visual Basic Professional edition, for example, may run in the background without any visible forms.
- The ActiveX component you call may have been created using the Remote automation features of Visual Basic, Enterprise edition, and may be running on another computer located at some distance from the user.
- If your program has loaded a Microsoft Excel workbook using the **GetObject** function, the workbook will not be visible when the user switches to Microsoft Excel. In fact, Microsoft Excel itself may not be visible, in which case the Switch To button does nothing.

In these situations, the default text and Switch To button are inappropriate and may confuse the user of your program.

The **OLEServerBusyMsgText** property allows you to replace the default Server Busy dialog box with an alternate message box. Setting **OLEServerBusyMsgText** to your own message string causes the default Server Busy dialog box to be replaced by a simple message box containing your message text, an OK button, and a Cancel button.

If **OLERequestPendingMsgText** is equal to an empty string (""), the default Server Busy dialog is displayed.

If the user presses the Cancel button of the default Server Busy dialog box or the alternate message box, the ActiveX error -2147418111 (&H80010001) is raised in the procedure that made the automation request.

Important When you know that an automation request may take more than a few seconds and you are using a hidden or remote ActiveX component, you should set an alternate message. For remote ActiveX components, the alternate message is recommended for all requests. Network traffic may occasionally cause even a very short ActiveX request to take several seconds.

OLEServerBusyMsgTitle Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproOLEServerBusyMsgTitleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEServerBusyMsgTitleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEServerBusyMsgTitleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEServerBusyMsgTitleS"}

Returns or sets the caption of the alternate "busy" message which is displayed when an ActiveX component rejects an automation request. Not available at design time.

Syntax

object.**OLEServerBusyMsgTitle** [= *string*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	A string expression that evaluates to the caption of the alternate message box for the ActiveX component busy condition.

Remarks

If the **OLEServerBusyMsgText** property has been set, the value of the **OLEServerBusyMsgTitle** property is used as the caption of the alternate busy message. The default value of the **OLEServerBusyMsgTitle** property is the current value of the **Title** property of the **App** object. This is the recommended setting.

If the **OLEServerBusyMsgText** property is set to an empty string (""), the **OLEServerBusyMsgTitle** property is ignored.

OLEServerBusyRaiseError Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproOLEServerBusyRaiseErrorC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEServerBusyRaiseErrorX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEServerBusyRaiseErrorA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEServerBusyRaiseErrorS"}

Determines whether a rejected automation request raises an error, instead of displaying a "busy" message. Not available at design time.

Syntax

object.**OLEServerBusyRaiseError** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that specifies whether an error is to be raised, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) An error is raised when the number of milliseconds specified by the **OLEServerBusyTimeout** property have elapsed.
False	Depending on the setting of the **OLEServerBusyMsgText** property, either the default Server Busy dialog box or an alternate busy message will be displayed.

Remarks

Raising an error when an ActiveX component rejects an automation request returns control to your program, which allows you to provide your own custom dialog box in place of either the default Server Busy dialog box or the alternate busy message.

The automation error that will be raised is -2147418111 (&H80010001).

OLEServerBusyTimeout Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproOLEServerBusyTimeoutC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEServerBusyTimeoutX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEServerBusyTimeoutA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEServerBusyTimeoutS"}

Returns or sets the number of milliseconds during which an automation request will continue to be retried, before a "server busy" message is displayed. Not available at design time.

Syntax

object.**OLEServerBusyTimeout** [= *milliseconds*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
milliseconds	A **Long** integer representing the number of milliseconds during which an automation request will be retried.

Remarks

The default value of this property is 10000 milliseconds (ten seconds).

Important This time-out value also affects documents you link or embed using the **OLE Container** control or the Toolbox. If you are using linked or embedded documents and you change this property before an automation request, it is a good idea to reset the value afterward.

EXEName Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproEXENameC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproEXENameX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproEXENameA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproEXENameS"}

Returns the root name of the executable file (without the extension) that is currently running. If running in the development environment, returns the name of the project.

Syntax

object.**EXEName**

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

Form Window Keys

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbkbdFormWinC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbkbdFormWinS"}

Use these key combinations in the Form window:

Press	To
F7	Open the Code window for the selected object.
CTRL+C	Copy the selected controls to the Clipboard.
CTRL+X	Cut the selected controls to the Clipboard.
DEL	Delete the selected controls without placing them on the Clipboard.
CTRL+V	Paste the Clipboard contents on the form.
CTRL+Z	Undo a deletion of controls.
TAB	Cycle forward through controls in tab order.
SHIFT+TAB	Cycle backward through controls in tab order.
CTRL+CLICK	Add or remove a control from the selection.
CLICK+DRAG	Select multiple controls.
CTRL+CLICK+DRAG	Add or remove controls from the current selection.
SHIFT+CLICK	Select multiple controls.
CTRL+E	Display the Menu Editor (design time only).
F4	Display the Properties window (design time only).
CTRL+J	Bring to front (affects overlapping controls at design time only).
CTRL+K	Send to back (affects overlapping controls at design time only).
CTRL+DOWN ARROW	Move the control down one grid unit (if the grid is turned on) or one pixel (if the grid is turned off).
CTRL+UP ARROW	Move the control up one grid unit (if the grid is turned on) or one pixel (if the grid is turned off).
CTRL+RIGHT ARROW	Move the control one grid unit (if the grid is turned one)or one pixel (if the grid is turned off) to the right.
CTRL+LEFT ARROW	Move the control one grid unit (if the grid is turned on) or one pixel (if the grid is turned off) to the left.

To deselect all controls, click the form. To select controls in a container, first deselect the container and then CTRL+CLICK+DRAG around the desired controls.

Menu Editor Keys

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbkbdMenuEditorC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbkbdMenuEditorS"}

Use these key combinations in the Menu Editor:

Press	To
TAB	Cycle forward through the boxes and buttons.
SHIFT+TAB	Cycle backward through the boxes and buttons.
ENTER	Cycle forward through menu items.
ALT+R	Move an item to a lower level in a hierarchical menu.
ALT+L	Move an item to a higher level in a hierarchical menu.
ALT+U	Move an item one line up.
ALT+B	Move an item one line down.

Use these key combinations when the Shortcut list box has the focus in the Menu Editor:

Press	To
F4	Open or close the list.
ALT+DOWN ARROW	Open or close the list.
ALT+UP ARROW	Open or close the list.
END	Move to the last item in the list.
HOME	Move to the first item in the list.

Toolbox Keys

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbkbdToolboxKeysC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbkbdToolboxKeysS"}

Use these key combinations in the Toolbox:

Press	To
ENTER	Place the selected control on the active form.
DOWN ARROW	Select the next tool down in the same column as the selected tool.
UP ARROW	Select the next tool up in the same column as the selected tool.
LEFT ARROW	Select the tool to the left of the selected tool.
RIGHT ARROW	Select the tool to the right of the selected tool.
TAB	Move through the Toolbox from left to right, one tool at a time.
SHIFT+TAB	Move up through the Toolbox from right to left, one tool at a time.
END	Select the last tool in the Toolbox.
HOME	Select the pointer tool.
ALT+F4	Close the Toolbox.

TaskVisible Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTaskVisibleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTaskVisibleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTaskVisibleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTaskVisibleS"}

Returns or sets a value that determines if the application appears in the Windows task list.

Syntax

object.**TaskVisible** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A boolean expression that determines if the application appears in the task list, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) The application appears in the Windows task list.
False	The application does not appear in the Windows task list.

Remarks

The **TaskVisible** property can only be set to **False** in applications that do not display an interface, such as OLE servers that do not contain or display **Form** objects. While the application displays an interface, the **TaskVisible** property is automatically set to **True**.

hInstance Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHInstanceC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHInstanceX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHInstanceA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHInstanceS"}

Returns a handle to the instance of the application.

Syntax

object.**hInstance**

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

Remarks

The **hInstance** property returns a long value.

When working with a project in the Visual Basic development environment, the **hInstance** property returns the instance handle of the Visual Basic instance.

Comments Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproCommentsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCommentsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCommentsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCommentsS"}

Returns or sets a string containing comments about the running application. Read only at run time.

Syntax

object.**Comments**

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

Remarks

You can set this property at design time in the Type box in the Make tab of the Project Properties dialog box.

CompanyName Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproCompanyNameC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCompanyNameX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCompanyNameA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCompanyNameS"}

Returns or sets a string value containing the name of the company or creator of the running application. Read only at run time.

Syntax

object.**CompanyName**

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

Remarks

You can set this property at design time in the Type box in the Make tab of the Project Properties dialog box.

FileDescription Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFileDescriptionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFileDescriptionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFileDescriptionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFileDescriptionS"}

Returns or sets a string value containing file description information about the running application. Read only at run time.

Syntax

object.**FileDescription**

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

Remarks

You can set this property at design time in the Type box in the Make tab of the Project Properties dialog box.

LegalCopyright Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLegalCopyrightC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLegalCopyrightX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLegalCopyrightA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLegalCopyrightS"}

Returns or sets a string value containing legal copyright information about the running application. Read only at run time.

Syntax

object.**LegalCopyright**

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

Remarks

You can set this property at design time in the Type box in the Make tab of the Project Properties dialog box.

LegalTrademarks Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLegalTrademarksC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLegalTrademarksX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLegalTrademarksA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLegalTrademarksS"}

Returns or sets a string value containing legal trademark information about the running application. Read only at run time.

Syntax

object.**LegalTrademarks**

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

Remarks

You can set this property at design time in the Type box in the Make tab of the Project Properties dialog box.

Major Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMajorC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMajorX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMajorA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMajorS"}

Returns or sets the major release number of the project. Read only at run time.

Syntax

object.**Major**

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

Remarks

The value of the **Major** property is in the range from 0 to 9999.

This property provides version information about the running application.

You can set this property at design time in the Major box in the Make tab of the Project Properties dialog box.

Minor Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMinorC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMinorX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMinorA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMinorS"}

Returns or sets the minor release number of the project. Read only at run time.

Syntax

object.**Minor**

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

Remarks

The value of the **Minor** property is in the range from 0 to 9999.

This property provides version information about the running application.

You can set this property at design time in the Minor box in the Make tab of the Project Properties dialog box.

ProductName Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproProductNameC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproProductNameX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproProductNameA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproProductNameS"}

Returns or sets a string value containing the product name of the running application. Read only at run time.

Syntax

object.**ProductName**

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

Remarks

You can set this property at design time in the Type box in the Make tab of the Project Properties dialog box.

Revision Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproRevisionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRevisionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRevisionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRevisionS"}

Returns or sets the revision version number of the project. Read only at run time.

Syntax

object.**Revision**

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

Remarks

The value of the **Revision** property is in the range from 0 to 9999.

This property provides version information about the running application.

You can set this property at design time in the Minor box in the Make tab of the Project Properties dialog box.

A default printer does not exist (Error 28663)

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

Check the system for installed printers using the **Printers** collection. If there are printers installed, set the **Printer** object to one of the printers in the **Printers** collection to define a default printer.

An attempt to subclass a ListBox failed due to insufficient memory (Error 20478)

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

The **CommonDialog** control uses specialized functions in the operating system to present the dialog boxes it displays. When this error occurs, the problem exists at the operating system level and shouldn't appear as the result of Visual Basic code that uses the control.

Cancel was selected (Error 32755)

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

This error occurs when the **CancelError** property is set to **True** and the user clicks the Cancel button in the dialog box. Use this error to execute different code when the Cancel button was selected. For example, in a File Open dialog box, you might respond to this error by exiting the current procedure instead of attempting to open a file.

Help call fail. Check Help properties (Error 32751)

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

Check to make sure the **HelpContextID** and **HelpFile** properties are set to proper values for the Help file used by the application.

Invalid filename (Error 20477)

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

The specified filename may not exist or may contain illegal characters.

Low on memory! Can't bring up the dialog! (Error 32752)

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

This error occurs when sufficient memory isn't available. Check for sufficient system memory as well as available system resources.

No fonts exist (Error 24574)

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

Before displaying the Choose Font dialog box, you must set one of the following flags:

- **ScreenFonts**
- **PrinterFonts**
- **Both**

No printer device drivers were found (Error 28664)

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

The system may not have printer drivers installed or the drivers may be missing.

The common dialog function failed during initialization (Error 32765)

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

This error often occurs when sufficient memory isn't available.

The common dialog function failed to find a specified string (Error 32761)

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

The **CommonDialog** control uses specialized functions in the operating system to present the dialog boxes it displays. When this error occurs, the problem exists at the operating system level and shouldn't appear as the result of Visual Basic code that uses the control.

The common dialog function failed to load a specified resource (Error 32760)

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

The **CommonDialog** control uses specialized functions in the operating system to present the dialog boxes it displays. When this error occurs, the problem exists at the operating system level and shouldn't appear as the result of Visual Basic code that uses the control.

The common dialog function failed to load a specified string (Error 32762)

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

The **CommonDialog** control uses specialized functions in the operating system to present the dialog boxes it displays. When this error occurs, the problem exists at the operating system level and shouldn't appear as the result of Visual Basic code that uses the control.

The common dialog function failed to lock a specified resource (Error 32759)

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

The **CommonDialog** control uses specialized functions in the operating system to present the dialog boxes it displays. When this error occurs, the problem exists at the operating system level and shouldn't appear as the result of Visual Basic code that uses the control.

The common dialog function failed to parse the strings in the [devices] section of the file WIN.INI (Error 28669)

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

This error occurs when the information in Win.ini can't be read by the **CommonDialog** control. For example, the Win.ini entry may contain the string "LDT1" instead of "LPT1" to identify a port.

The common dialog function was unable to allocate memory for internal data structures (Error 32758)

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

This error is caused by insufficient memory.

The common dialog function was unable to lock the memory associated with a handle (Error 32757)

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

The **CommonDialog** control uses specialized functions in the operating system to present the dialog boxes it displays. When this error occurs, the problem exists at the operating system level and shouldn't appear as the result of Visual Basic code that uses the control.

The data in the DEVMODE and DEVNAMES data structures describes two different printers (Error 28662)

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

The **CommonDialog** control uses specialized functions in the operating system to present the dialog boxes it displays. When this error occurs, the problem exists at the operating system level and shouldn't appear as the result of Visual Basic code that uses the control.

The ENABLETEMPLATE flag was set in the Flags member of a common dialog data structure but the application failed to provide a corresponding instance handle (Error 32763)

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

The **CommonDialog** control uses specialized functions in the operating system to present the dialog boxes it displays. When this error occurs, the problem exists at the operating system level and shouldn't appear as the result of Visual Basic code that uses the control.

The ENABLETEMPLATE flag was set in the Flags member of a common dialog data structure but the application failed to provide a corresponding template (32764)

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

The **CommonDialog** control uses specialized functions in the operating system to present the dialog boxes it displays. When this error occurs, the problem exists at the operating system level and shouldn't appear as the result of Visual Basic code that uses the control.

The FileName buffer is too small to store the selected file name(s) (Error 20476)

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

This error occurs when the **MaxFileSize** property setting is too small to allocate enough space in memory for the selected file name(s). Increase the setting of the **MaxFileSize** property.

The PD_RETURNDEFAULT flag was set in the Flags member of the PRINTDLG data structure but either the hDevMode or hDevNames field was nonzero (Error 28668)

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

The **CommonDialog** control uses specialized functions in the operating system to present the dialog boxes it displays. When this error occurs, the problem exists at the operating system level and shouldn't appear as the result of Visual Basic code that uses the control.

The PrintDlg function failed during initialization (Error 28665)

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

This error is usually caused by insufficient memory.

The PrintDlg function failed to load the specified printer's device driver (Error 28667)

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

This error could be caused by a defective printer driver or a corruption in system memory.

The **CommonDialog** control uses specialized functions in the operating system to present the dialog boxes it displays. When this error occurs, the problem exists at the operating system level and shouldn't appear as the result of Visual Basic code that uses the control.

The PrintDlg function failed when it attempted to create an information context (Error 28661)

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

The amount of available system resources may be too low.

The printer device driver failed to initialize a DEVMODE data structure (Error 28666)

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

The **CommonDialog** control uses specialized functions in the operating system to present the dialog boxes it displays. When this error occurs, the problem exists at the operating system level and shouldn't appear as the result of Visual Basic code that uses the control.

The [devices] section of the file WIN.INI did not contain an entry for the requested printer (Error 28660)

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

The **CommonDialog** control uses specialized functions in the operating system to present the dialog boxes it displays. When this error occurs, the problem exists at the operating system level and shouldn't appear as the result of Visual Basic code that uses the control.

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

'Item' is in binary format and cannot be loaded into VB5

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

Binary format files created with older versions of Visual Basic can't be loaded into Visual Basic 5.0. Only Visual Basic version 2.0 and 3.0 support converting binary format files. If you have a copy of Visual Basic 2.0 or 3.0, you can load the file and save it as text. Then you can load the file in Visual Basic 5.0.

Can't display system information

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

Visual Basic can't display system information when you choose System Info from the About Microsoft Visual Basic dialog box. Your system may not have enough memory or a required file may be corrupted or missing.

Separator may not be checked or disabled, or have a shortcut key

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

While working with the Menu Editor tool, you tried to set a separator menu item's **Checked** property to **True** or its **Enabled** property to **False** or you tried to assign it a shortcut key. A separator menu item can't have any of these property settings, but you can set its **Visible** property to **False**.

Only Top Level Menus can have non-zero NegotiatePosition

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

You tried to set the **NegotiatePosition** property for a menu item other than a top-level menu.

Project too large to make into an .EXE

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

This project exceeded the absolute size for an .EXE file that the operating system can handle. You must reduce the size of the project or break it up into pieces.

Version numbers must be in the range 0 to 9999

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

You tried to assign a negative number, a non-numeric character, or a number greater than 9999, as the Major, Minor, or Revision component of the Version number for a project in the Make tab of the Project Properties dialog box. Use a number between 0 and 9999 inclusive.

Only one resource file allowed

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

You chose Add File from the Project menu and tried to add a second .RES file to your project, or the .VBP file lists more than one .RES file. You can only have one Windows resource file in a Visual Basic project.

Duplicate resources with same type and name

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

The Windows resource file you are trying to use in your project contains two or more resources with the same type and name, or the resource file has a resource that Visual Basic automatically creates.

Use another resource file or recreate the invalid resource file and delete one of the duplicate resources.

Programmatic ID string too long 'item'

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

The ProgID string of the ActiveX component can't be longer than 39 characters. The ProgID is created by concatenating the project name with the class module name.

ActiveX component could not be created

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

An ActiveX component or instance of it couldn't be created with the **CreateObject** function, or you tried to compile the project, but Visual Basic couldn't create the file.

The Public property for Class 'item' cannot be set to True. You do not have an appropriate license to use this functionality in the design environment.

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

You can only make Public class modules (ActiveX components accessible from outside the project) with the Professional and Enterprise editions of Visual Basic 5.0.

The file 'item' does not contain information needed to create a compatible ActiveX component

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

The filename you entered for the Compatible ActiveX Component, in the Component tab of the Project Options dialog box, isn't a Visual Basic-created .EXE with Public class modules.

Do you want to upgrade all 'item1' objects in this project to 'item2' objects?

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

The class of the specified object has changed. Choose Yes to upgrade all instances of the specified object in this project to reflect the changes made to the class, or choose No if you don't want to upgrade.

The class name of the newly upgraded 'item1' objects is 'item2'. Since this is different from their previous name of 'item3', you must change any declarations of this object that you have in code.

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

The class name of the specified object you upgraded has changed. You must change any declarations of the specified object in your code to refer to the new class name.

Could not find Data Access Library. Cannot Create reference.

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

Visual Basic tried to create a reference to the Microsoft DAO Object Library through the **Data** control, but failed. Verify that all system registry keys are valid and, if they are not, run Visual Basic setup again.

Can't exit Windows while an OLE object is active for editing

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

You tried to exit Windows while an **OLE** object was active for editing. Deactivate the **OLE** object before quitting Windows.

Other applications are currently accessing an object in your program. Ending the program now could cause errors in those programs. End program at this time?

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

An automation object in the Visual Basic program you are currently running is being accessed by at least one other application. Terminating your program could cause an error in the other application. Choose Yes to quit your program or choose No to continue.

One or more of the properties in 'item' was bad. Some or all of the properties might not be set correctly.

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

The specified .VBP file has one or more invalid property settings. This error occurs only in regard to a project's .VBP file.

'Item' could not be registered

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

The specified file couldn't be registered in the system registry. The error is related to type libraries used by components of the Visual Basic development environment and indicates that the specified file's entry in the system registry is corrupted or that the DLL itself is missing or corrupted.

'Item' could not be loaded. Remove it from the list of available Add-Ins?

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

Visual Basic couldn't load the add-in that you tried to select from the Available Add-Ins list in the Add-In Manager dialog box. Choose Yes to remove it from the list or choose No to leave it on the list. In either case, you won't be able to load it.

Visual Basic cannot load 'item' because it is not in the system registry. Remove it from the list of available Add-Ins?

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

Visual Basic couldn't load the add-in that you tried to select from the Available Add-Ins list in the Add-In Manager dialog box, because it was not registered properly or is no longer registered in the system registry.

Visual Basic can't upgrade the custom controls to those provided in the 'item' library. Remove this library from the list of available upgrades?

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

Visual Basic was unable to convert one or more VBX controls to ActiveX controls. Choose Yes to remove the library reference from the list so that VB won't try to upgrade the specified VBX custom controls again, or choose No to leave the library in the list.

Not enough memory to run; quitting

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

Visual Basic couldn't obtain enough memory to run. Close other applications and try again.

Not enough memory to load file

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

Visual Basic couldn't obtain enough memory to load the file. Close other applications and try again.

Unexpected error; quitting

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

An unexpected error occurred, and Visual Basic was unable to continue. This may be a hardware problem or an effect of other software in your system.

Wrong version of run-time DLL

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

Visual Basic can't find VBRUN500.DLL, or it is finding the wrong version of one or more of the ActiveX control .ocx files. Make sure your system contains only one copy of these files (the latest version) in your \SYSTEM or \SYSTEM32 directory.

Main can't be module, type, project, or form name.

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

You have the startup form set to Sub Main, and you have a module, form, project, or type named "Main" which prevents the Sub with the same name from being called.

Rename the module, form, project, or type.

Can't put check mark here.

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

This is a menu design error indicating that you tried to select Checked on a menu item that does not support check marks. The following situations can cause this error:

- You tried to select Checked in the Menu Editor for a top-level menu item.
- You selected Checked for some menu item which has a submenu such as a cascading sub-menu.
- You chose some menu such as a MDI Window list menu that cannot be checked.

Wrong version of operating system; requires Windows NT 3.51 (build 'item1' or above), or Windows 95 (build 'item2' or above)

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

To run this version of Visual Basic, you must be running the specified build of Windows 95 or Windows NT version 3.51.

'Item' is a Read-Only file

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

You asked to save to a file that is read-only. Read-only files are shown in the Project Explorer as  . You can't save to read-only files. Use Save As instead.

Can't have more than one Window List menu

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

You clicked the Window List check box for more than one menu item and then chose OK to close the dialog box. For a particular form, only one menu can have the Window List check box checked. To do this, use the Menu Editor.

Controls without the align property cannot be placed directly on the MDI form

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

You tried to double-click a control to place it on an MDI form, but the control doesn't support the standard **Align** property or is not an invisible at run-time control. The **PictureBox** and **Timer** controls are the only intrinsic controls that can be placed on an MDI form.

Duplicate procedure name

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

You have defined (or are attempting to define) more than one version of a procedure with the same name. Rename one of the procedures. Ensure that duplicate **Declare** statements for the same procedure have the same **Alias** clause.

Separator may not be the Window List menu

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

The window menu has to be a menu item that can support being a pop-up menu (one that can have subitems). A separator can't have subitems.

Valid values are whole numbers from 1 to 32

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

You attempted to set the TabStop Width beyond the permitted range. A number between 1 and 32 (inclusive) is acceptable for this field.

Valid values are whole numbers from 24 to 1188

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

You attempted to set the **Grid Width** or **Grid Height** beyond the permitted range. A number between 24 and 1188 (inclusive) is acceptable for this field.

Errors during load. Refer to 'item' for details

Something unexpected appeared in the ASCII form file. Visual Basic created a log file to provide more detail about the errors. You should examine the log file to determine the severity of the problem. Sometimes you can safely ignore the errors (for example, `Version number missing or invalid...`). Other times, however, the errors could cause the form not to run as expected (for example, `Class MyClass in control  MyControl was not a loaded control class`).

String value too long to process; form load aborted

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

A string embedded in the form being loaded was too long to process.

Version number missing or invalid; Visual Basic 5.0 assumed

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

Either the version signature was not found, or the specified version isn't recognized. Make sure that the first line is not blank or a comment. It should read `VERSION 5.00` in the ASCII form.

Line 'item1': The file 'item2' could not be loaded.

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

Syntax errors are preventing Visual Basic from parsing and loading a file, or form name conflicts prevent loading of an ASCII form. The form won't be loaded and the form name won't be displayed in the Project Explorer.

Make sure that the file causing this error is a valid ASCII form in the correct format and that no conflicts exist among the different forms in the project. The correct ASCII form format is:

```
VERSION 5.00
Begin VB.Form <formname>

   Form Properties
   Begin VB.<controltype> <controlname>
      prop = value
      prop = value
      [...]

      Control Properties
      Begin VB.<control type> <control name>
         prop = value
         [...]
         Begin ...
            [...]
         End
      End
   End
End
ATTRIBUTE = "<form name>"
[...]
```

'Item' could not be loaded

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

Syntax errors are preventing Visual Basic from parsing and loading a file, or form name conflicts prevent loading of an ASCII form. The form won't be loaded and the form name won't be displayed in the Project Explorer.

Make sure that the file causing this error is a valid ASCII form in the correct format and that no conflicts exist among the different forms in the project. The correct ASCII form format is:

```
VERSION 5.00
Begin VB.Form <formname>

   Form Properties
   Begin VB.<controltype> <controlname>
      prop = value
      prop = value
      [...]

      Control Properties
      Begin VB.<control type> <control name>
         prop = value
         [...]
         Begin ...
            [...]
         End
      End
   End
End
ATTRIBUTE = "<form name>"
[...]
```

Can't remove control or reference; in use

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

The ActiveX control or reference that you tried to remove is being used by one of the forms in the project. First delete the control or referenced object from the form and then deselect it from the list.

Must have startup form or Sub Main()

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

No form in the current project is designated as the startup form, and the current project doesn't have a **Sub** procedure named `Main` in any module. You must have one or the other to run a Visual Basic application.

Can't save file to TEMP

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

The directory established by your TEMP environment variable can't be written to. Make sure that your TEMP environment variable is set to a valid directory and that the disk isn't full.

Can't find Windows Help .EXE file

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

The Windows Help application isn't available. If Winhelp.Exe is on your machine, make sure it is on your path. If it isn't on your machine, run Microsoft Windows 95 or Microsoft Windows NT setup to install it.

Invalid procedure name

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

A procedure name can't be a restricted keyword, must start with a letter, must be unique within the same scope, and can be a maximum of 255 characters — including letters, numbers, and underscores (_). **Function** procedure names can include a type declaration character, but **Sub** procedure names cannot.

Array already has control at index 'item'

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

Possible causes:

- You tried to change the index value to a number that is already used by this control array. To determine which indices have already been assigned, select the **Index** property in the Properties window for each control array element.
- You tried to change the **Name** property of a control in an array to the name of a control array already using this index value.

Not a legal object name: 'item'

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

Form and control names must start with a letter and can be a maximum of 40 characters — including letters, numbers, and underscores (_).

Note that the **Name** property of a form or control is different from the **Label** properties — **Caption**, **Text**, and **Value** — which label or display the contents of a control at run time. These properties can be restricted keywords, can begin with a number, and can contain nonalphanumeric characters.

Must specify which item(s) to print

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

You have not specified what to print. Check at least one of the Form or Code check boxes in the Print dialog box after choosing Print from the File menu.

Can't assign shortcut key to a top level menu

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

You are trying to assign a key combination to a parent or top-level menu item that isn't a command. Only commands can have shortcut keys assigned to them.

Shortcut key already assigned

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

You tried to assign a shortcut key that has already been assigned to another menu item. Assign a different shortcut key.

At least one submenu item must be visible

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

The **Visible** property can't be set to **False** for the last remaining visible submenu item. You can't have a parent menu with no visible submenu items.

Can't quit at this time

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

You can't quit Windows while Visual Basic is in run or break mode. You also can't quit Windows while a dialog box or message box is displayed.

Invalid command line argument 'item'

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

There is an invalid argument in the command line used to invoke Visual Basic. Search Help for a list of command line arguments.

You already have a control named 'item'. Do you want to create a control array?

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

You've tried to set the **Name** property of a control or paste a new control onto the form, and you already have a control with the indicated name.

If you would like to create a control array, choose Yes. The newly created control array element will have an Index value one higher than the control previously created with this name. If you don't want to create a control array, choose No. If you're pasting a control, Visual Basic will generate the name for the new control.

Error loading 'item'. An error was encountered loading a property. Continue?

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

You may have tried to load a form with controls whose names conflict with forms already in the project. For example, loading Form2 that contains a Form1 control triggers this error.

Error loading 'item'. A control could not be loaded due to load error. Continue?

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

This error message appears after another error has occurred. Once you've taken the appropriate action for that error, you will see this error message. To load the control anyway, choose Yes; to cancel the loading, choose No.

'Item' already exists in project

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

The file you specified is already part of the project. You can't add the same file to a project more than once. You can't save a file with the same name as another file in the project.

'Item' has an old file format. When saved it will be saved in a newer format.

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

This file was created with an earlier version of Visual Basic. When you save it, it will be saved in the Visual Basic 5.0 file format.

Out of memory (31001)

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

Could not allocate or access enough memory or disk space for the specified operation.

Can't copy (Error 31009)

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

You tried to copy an object to the **Clipboard** (set **Action** = 4), but the object is corrupted or is no longer valid.

Can't open Clipboard (Error 31003)

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

You tried to copy (**Action** = 4) or paste (**Action** = 5) an object, but the **Clipboard** is not currently available. Find the application that has the **Clipboard** open, and close it.

Can't paste (Error 31007)

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

The **Clipboard** doesn't contain a valid OLE object.

This error also occurs when you copy an OLE object to the **Clipboard** and then try to paste the object back to the same **OLE** container control.

The **PasteOK** property returns **True** when an object can be pasted from the **Clipboard**.

Note that when you copy an OLE object to the **Clipboard** (set **Action** = 4) and then delete the object (set **Action** = 10), the object on the **Clipboard** is also deleted.

Class is not set (Error 31018)

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

When setting **Action** = 0 (**CreateEmbed** method) if you don't specify a source document (**SourceDoc** property), the **Class** property must be set to the name of a class available on your system.

To display a list of the available class names at design time, click the **OLE** container control with the right mouse button and choose the Insert Object command.

Dialog already in use (Error 31029)

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

Only one OLE dialog box (either Insert Object or Paste Special) can be displayed at a time.

Error loading from file (Error 31037)

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

An error occurred while attempting to read the specified file (set **Action** = 12). Possible causes:

- The **FileNumber** property is invalid.
- The file wasn't opened in **Binary** mode.
- The file wasn't saved properly (set **Action** = 11).
- The file is corrupted.
- The file position isn't located at the beginning of a valid OLE object.

Error saving to file (Error 31036)

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

Visual Basic can't write the object to the specified file (set **Action** = 11, or 18). Possible causes:

- The **FileNumber** property is invalid.
- The specified file wasn't opened in **Binary** mode.
- There isn't enough disk space.

Invalid Action (Error 31021)

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

The **Action** property has been set to an invalid value.

The values 2, 3, 8, 13, and 16 are reserved for future use and are not valid with this version of the **OLE** container control.

Incorrect Clipboard format (Error 31035)

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

The **OleTypeAllowed** property doesn't match the type of object on the **Clipboard**.

Invalid format (Error 31017)

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

You attempted to set or get data with the **Data** or **DataText** property using an invalid data format or a format the object doesn't support.

To determine the set of valid formats for an OLE object, use the **ObjectAcceptFormats**, **ObjectAcceptFormatsCount**, **ObjectGetFormats**, and **ObjectGetFormatsCount** properties.

Invalid or unknown Class (Error 31023)

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

The specified **Class** property isn't valid and the **SourceDoc** property isn't specified.

To display a list of the available class names on your system, at design time click the **OLE** container control with the right mouse button and choose the Insert Object command.

Use extra care when setting the **Class** property at run time as some class names may be case-sensitive.

Invalid property value (Error 31008)

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

You attempted to assign an invalid value to a property.

This error also occurs when you are performing an action that requires several properties to be set, and one of the properties is set to an invalid value.

Invalid source for link (Error 31031)

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

You attempted to create a linked object (set **Action** = 1) and the file specified by the **SourceDoc** property couldn't be found.

Invalid Verb index (Error 31034)

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

You tried to activate an object (set **Action** = 7), and the **Verb** property was set to an invalid value.

Use the **ObjectVerbs** and **ObjectVerbsCount** properties to determine the list of verbs an object supports.

Source name is too long (Error 31026)

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

The combined number of characters in the **SourceDoc** and **SourceItem** strings can't exceed 256.

No object (Error 31004)

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

You tried to perform an action on an object that doesn't exist.

Object not running (Error 31028)

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

You can't perform the following actions on an object unless the application that created the object is currently running:

- Copy an object to the **Clipboard** (set **Action** = 4).
- Set or read the **Data** property.
- Set or read the **DataText** property.
- Read the **ObjectAcceptFormats** property.
- Read the **ObjectAcceptFormatsCount** property.
- Read the **ObjectGetFormats** property.
- Read the **ObjectGetFormatsCount** property.

Use the **AppIsRunning** property to determine if the application that created an object is running.

If the object exists, you activate it (and therefore invoke the application that created it) by setting **Action** = 7 (Activate). To activate an object without displaying the application that created it, first set the **Verb** property to – 3.

Source Document is not set (Error 31019)

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

When creating a linked object, or creating an object from a file (set **Action** = 1), the **SourceDoc** property must be set to a valid file name.

At design time, select the **SourceDoc** property in the Properties window and click the three dots in the settings box to browse your disk for a valid filename.

Unable to access source document (Error 31039)

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

You attempted to create an embedded object, but the file specified by the **SourceDoc** property isn't a valid file.

Unable to activate object (Error 31027)

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

The object's source document can't be loaded, or the application that created the object isn't available.

This error occurs when you try to activate a linked object (set **Action** = 7) and the file specified in the **SourceDoc** property has been deleted, or no longer exists.

This error also occurs when you activate an object (set **Action** = 7), and the action specified by the **Verb** property isn't valid. Some applications that provide objects may support different verbs, depending on the state of the application. All the verbs supported by an application are listed in the **ObjectVerbs** property list. However, some verbs may not be valid for the application's current state.

Unable to close object (Error 31006)

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

The object can't be closed (**Action** = 10) in its current state.

Unable to create embedded object (Error 31032)

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

The application that is creating the object can't create the object as specified in the **SourceDoc** property.

For example, this error occurs if you try to embed a spreadsheet object and **SourceDoc** specifies a spreadsheet that is too large to be loaded by the spreadsheet application.

Unable to create link (Error 31024)

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

The application that is creating the object can't create the linked object as specified in the **SourceDoc** and **SourceItem** properties.

For example, this error occurs if you try to link a spreadsheet object and **SourceDoc** specifies a spreadsheet that is too large to be loaded by the spreadsheet application.

Unable to fetch link source name (Error 31033)

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

The **OLE** container control can't determine the name of the source document for the object you want to create.

For example, when you attempt to paste a linked object from the **Clipboard** (set **Action** = 5), the **OLE** container control must determine the name of the source document based on the information provided by the **Clipboard**. If the **OLE** container control can't determine the name of the document to link to, this error occurs.

Could not access the desired Column (Error 537)

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

The control could not access the desired column. This error has the following cause and solution:

- The bound control has attempted to access data from a column that is not in the result set.

  This error occurs when a control incorrectly accesses data from a **Data** control. Contact the control vendor for more information.

Method not applicable in this context (Error 444)

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

You cannot use this method in the current event procedure.

No timer available (Error 260)

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

The number of **Timer** controls in a project, and the amount of memory each Timer object requires are limited. This error has the following causes and solutions:

- There are too many active **Timer** controls. The maximum number varies from environment to environment.

  Remove one or more **Timer** controls from the project.
- There isn't enough memory to load another **Timer** control.

  Try to free up some memory by closing other applications.

No foreign application responded to a DDE initiate (Error 282)

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

Visual Basic could not find an application and topic corresponding to the application name and topic in the **LinkTopic** property. This error has the following causes and solutions:

- The application specified in **LinkTopic** isn't running.
  Ensure that the specified application is running.
- The application is running, but doesn't recognize the topic of the link.
  Ensure that the specified application actually has a topic with the name specified in **LinkTopic**.

Foreign application won't perform DDE method or operation (Error 285)

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

An application refused to perform the DDE method or operation you attempted. This error has the following causes and solutions:

- You supplied data or commands that the other application did not recognize.

  Check the application's documentation to see what data or commands it recognizes.
- The **LinkItem** property isn't set to an item that the other application recognizes as valid for the topic of the conversation.

  Check the application's documentation to see what items it recognizes.

Timeout while waiting for DDE response (Error 286)

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

The other application in a DDE conversation did not respond within the time specified by the **LinkTimeout** property. This error has the following causes and solutions:

- The other application isn't responding because it is waiting for a response from the user.

  Switch to that application, and close the dialog box or take an action corresponding to the message it displays.
- The **LinkTimeout** property is set to a value that is too low.

  Try increasing the value.
- The other application is too busy to respond to DDE messages.

  Try calling the **DoEvents** function before performing this DDE operation.

User pressed Escape key during DDE operation (Error 287)

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

You pressed the ESC key while waiting for a DDE operation to be completed. This error has the following cause and solution:

- The other application in the DDE conversation is taking too long to respond.
  Try setting the **LinkTimeout** property to a lower value.

Destination is busy (Error 288)

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

The other application in the DDE conversation can't perform a DDE operation. This error has the following cause and solution:

- The other application is busy responding to other Windows events.
  Try calling the **DoEvents** function and attempt the DDE operation again.

Data in wrong format (Error 290)

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

An application in a DDE conversation supplied data in an unexpected format. It may not be performing DDE correctly. This error has the following causes and solutions:

- The application is supplying data in a format that Visual Basic doesn't recognize.
  Try initiating the conversation with a different topic.
- The application is supplying text data to a **PictureBox** or picture data to a **TextBox**.
  Try initiating the conversation with a different control.

DDE Method invoked with no channel open (Error 293)

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

A DDE method (**LinkExecute**, **LinkPoke**, **LinkRequest**, or **LinkSend**) was performed on a control that isn't involved in a valid DDE conversation. This error has the following causes and solutions:

- Changing the **LinkTopic** property terminates an existing DDE conversation, but doesn't automatically establish a new conversation.

  After changing the **LinkTopic** property for a control, you must set the **LinkMode** property of the form to 1 (Automatic) or 2 (Manual) before executing a DDE method on this control.
- You executed a DDE method on a control with **LinkMode** set to 0 (None).

  Set **LinkMode** to 1 (Automatic) or 2 (Manual) and try again.

Invalid DDE Link format (Error 294)

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

The other application in a DDE conversation passed data in an invalid format. This error has the following cause and solution:

- The application passed data in the **vbCFLink Clipboard** format but it isn't valid link data.
  Fix the data in the other application so that it conforms to the **vbCFLink Clipboard** format.

Message queue filled; DDE message lost (Error 295)

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

Visual Basic can't keep up with the number of DDE operations attempted. This error has the following causes and solutions:

- Too many DDE conversations are running.

  Try terminating one or more DDE conversations.
- Too much code in event procedures is executing because of incoming DDE data.

  Reduce the amount of code being called as a result of DDE changes, or try calling the **DoEvents** function.

PasteLink already performed on this control (Error 296)

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

You have already performed a Paste Link on this control. This error has the following cause and solution:

- You are trying to paste a new link while a prior link is still in effect.

  To paste a new link, first set the **LinkMode** property of this control to 0 (None), and then use the Paste Link command.

Can't set LinkMode; invalid LinkTopic (Error 297)

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

You tried and failed to set the **LinkMode** property of a control. This error has the following cause and solution:

- You did not specify a **LinkTopic** property for the control.
  First specify a valid **LinkTopic** property.

System DLL 'dll' could not be loaded (Error 298)

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

A .DLL file provided by the operating system; for example, Ddeml.dll, Version.dll, or Winspool.drv couldn't be found. This error has the following causes and solutions:

- The file isn't on the proper path.
  Ensure that the DLL is on the Windows System path.
- The DLL is corrupted or was deleted.
  Reload the DLL.
- There isn't enough memory or swap space.
  Try to free up some memory by closing other applications.

Can't use character device names in file names: 'item' (Error 320)

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

From within Visual Basic, you can't give a file the same name as a character device driver. This error has the following cause and solution:

- You tried to use a filename such as AUX, CON, COM1, COM2, LPT1, LPT2, LPT3, LPT4, or NUL. Give the file another name.

Invalid file format (Error 321)

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

A Visual Basic form file is damaged. This error has the following causes and solutions:

- The form has a damaged ActiveX control.

  Try replacing the ActiveX control on the form.
- The number of properties in the current version of the ActiveX control don't match the number expected.

  Try replacing the ActiveX control with an earlier or later version.

'Item' is not a valid resource file (Error 325)

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

A resource file in your project doesn't follow the standard format for a Windows resource file or there is a mismatch between the resource file and the Visual Basic project. This error has the following causes and solutions:

- You tried to use the **LoadResString**, **LoadResPicture**, or **LoadResData** methods on a resource file with an invalid format.

  Recreate the resource source and resource files and adhere to the syntax documented in the Windows Software Development Kit.
- You tried to make an .EXE file from a project that includes a resource file with an invalid format.

  Remove the invalid resource file from the project.
- You tried to use the Project Add File command to add a 16-bit resource file to a project.

Resource with identifier 'item' not found (Error 326)

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

The resource with the specified ID doesn't exist in the resource file. This error has the following cause and solution:

- You tried to use the **LoadResString**, **LoadResPicture**, or **LoadResData** methods to load a resource whose ID doesn't exist in the resource file associated with this project or .EXE file.

  Change the reference to the resource ID to one that does exist, or remove the resource file from the project and add a resource file that contains the specified resource ID.

Could not access system registry (Error 335)

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

An attempt to read from or write to the system registry failed.

ActiveX component not correctly registered (Error 336)

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

The ActiveX component has not been properly registered in the system registry.

Object server not found (Error 337)

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

The required EXE or DLL file can't be found.

Object server did not correctly run (Error 338)

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

The ActiveX component's EXE file failed to run correctly. There may be a problem with the information in the registry.

Control array element 'item' doesn't exist (Error 340)

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

You used an invalid index value to refer to an element in a control array. This error has the following cause and solution:

- No control with the index value you referred to is part of the existing control array.

  Change the **Index** property setting for one of the existing elements to the value you referred to. Or use the **Load** statement to add a control to the array with an index equal to the value and then refer to the index value again.

Invalid control array index (Error 341)

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

You used an invalid index value to refer to an element in a control array. This error has the following cause and solution:

- The index value you referred to in your code is a negative number.
  Change the index value to refer to one of the existing elements of the control array.

Not enough room to allocate control array 'item' (Error 342)

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

There isn't enough memory to create all the elements of a control array. This error has the following causes and solutions:

- An element in a control array with discontiguous index values for its elements, such as 0, 2, 4, was assigned too large an index value at design time.

  Change the **Index** property's setting to a smaller value so that Visual Basic doesn't run out of memory.
- You used the **Load** statement to add an element with too large an index value to a control array.

  Change the index value in your code to a smaller value.

Object not an array (Error 343)

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

A control that isn't part of a control array is referred to as if it were. This error has the following cause and solution:

- You tried to refer to an index value for a control that isn't part of an array (for example, `Command1(3).Caption` and `Command1.Text)`.

  Delete the reference to the index value, or redefine the control as an element in a control array.

Must specify index for object array (Error 344)

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

A control that isn't part of a control array is referred to as if it were. This error has the following causes and solutions:

- You referred to an index value for a control that isn't a part of a control array.

  Remove the reference to the index value.
- You tried to add a control at run time with the **Load** statement but no control array with that **Name** property exists.

  Create the control you are referring to as a control array at design time.

Reached limit: cannot create any more controls on this form (Error 345)

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

No more than 255 individual controls are allowed for each form. This error has the following cause and solution:

- At design time you tried to add a control to a form that resulted in a total of 256 individual controls for the form (with the form itself counting as one control).

  Delete one or more controls from the form or redefine one or more of your controls as part of a control array.

Object already loaded (Error 360)

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

The control in the control array has already been loaded. This error has the following cause and solution:

- You tried to add a control to a control array at run time with the **Load** statement but the index value you referred to already exists.

  Change the index reference to a new value or check if your code is executing the same **Load** statement, with the same index value reference, more than once.

Can't load or unload this object (Error 361)

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

A **Load** or **Unload** statement has referenced an invalid object or control. This error has the following causes and solutions:

- You tried to load or unload an object that isn't a control or form, such as **Screen**, **Printer**, or **Clipboard**.

  Delete the erroneous statement from your code.
- You tried to load or unload an existing control that isn't part of a control array. For example, assuming that a **TextBox** with the **Name** property Text1 exists, `Load Text1` will cause this error.

  Delete the erroneous statement from your code or change the reference to a control in a control array.
- You tried to unload a **Menu** control in the Click event of its parent menu.

  Unload the **Menu** control with some other procedure.
- You tried to unload the last visible menu item of a **Menu** control.

  Check the setting of the **Visible** property for the other menu items in the control array before trying to unload a menu item or delete the erroneous statement from your code.

Can't unload controls created at design time (Error 362)

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

Only a control array element loaded at run time can be unloaded. This error has the following cause and solution:

- You used the **Unload** statement to refer to a control array element that you created at design time.

  Change the index value of the control you want to unload to a control array element that was loaded at run time. Or you can hide any control by setting its **Visible** property to **False**.

ActiveX control 'item' not found (Error 363)

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

The form being loaded contains an ActiveX control that isn't part of the current project. This error has the following causes and solutions:

- You may have manually edited the project's .VBP file to add a form containing an ActiveX control that isn't already part of the project.

  After the project loads, select Components from the Project menu and add the ActiveX control to the project.
- You may have manually edited the project's .VBP file to add a form containing an ActiveX control that is an older version than the ActiveX control that is already part of the project.

  After the project loads, delete the older version from the form and put the newer version of the control on the form.

Object was unloaded (Error 364)

A form was unloaded from its own Form_Load procedure. This error has the following cause and solution:

- A form with an **Unload** statement in its Form_Load procedure was implicitly loaded. For example, the following will implicitly load Form2 if it isn't already loaded: `Form2.BackColor = Form1.BackColor`.

  Remove the **Unload** statement from the Form_Load procedure.

Unable to unload within this context (Error 365)

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

In some situations you are not allowed to unload a form or a control on a form. This error has the following causes and solutions:

- There is an **Unload** statement in the Paint event for the form or for a control on the form that has the Paint event.

  Remove the **Unload** statement from the Paint event.
- There is an **Unload** statement in the Change, Click, or DropDown events of a **ComboBox**.

  Remove the **Unload** statement from the event.
- There is an **Unload** statement in the Scroll event of an **HScrollBar** or **VScrollBar** control.

  Remove the **Unload** statement from the event.
- There is an **Unload** statement in the Resize event of a **Data**, **Form**, **MDIForm**, or **PictureBox** control.

  Remove the **Unload** statement from the event.
- There is an **Unload** statement in the Resize event of an **MDIForm** that is trying to unload an MDI child form.

  Remove the **Unload** statement from the event.
- There is an **Unload** statement in the RePosition or Validate event of a **Data** control.

  Remove the **Unload** statement from the event.
- There is an **Unload** statement in the ObjectMove event of an **OLE Container** control.

  Remove the **Unload** statement from the event.

No MDI form available to load (Error 366)

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

There is no **MDIForm** in your project to load. This error has the following cause and solution:

- You tried to load a form whose **MDIChild** property is set to **True**, but at run time there was no **MDIForm** in your project to load.

  Add a **MDIForm** to your project or change the **MDIChild** property setting to **False**.

The file 'item' is out of date. This program requires a newer version (Error 368)

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

An ActiveX control was found to be out of date. This error has the following cause and solution:

- The number of properties in the current version of the ActiveX control doesn't match the number expected.

  Replace the ActiveX control with a newer version.

Invalid property value (Error 380)

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

An inappropriate value is assigned to a property. This error has the following cause and solution:

- You tried to set one of the properties of an object or control to a value outside its permissible range. Change the property's value to a valid setting. For example, the **MousePointer** property must be set to an integer from 0 to 15 or 99.

Invalid property array index (Error 381)

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

An inappropriate property array index value is being used. This error has the following cause and solution:

- You tried to set a property array index to a value outside its permissible range.

  Change the index value of the property array to a valid setting. For example, the index value of the **List** property for a **ListBox** must be from 0 to 32,766.

'Item' property cannot be set at run time (Error 382)

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

The property is read-only at run time. This error has the following cause and solution:

- You tried to set or change a property whose value can only be set at design time.

  Remove the reference to the property from your code or change the reference to only return the value of the property at run time.

'Item' property is read-only (Error 383)

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

The property is read-only at both design time and run time. This error has the following cause and solution:

- You tried to set or change a property whose value can only be read.

  Remove the reference to the property from your code or change the reference to only return the value of the property at run time.

A form can't be moved or sized while minimized or maximized (Error 384)

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

The **Left**, **Top**, **Height**, and **Width** properties can't be changed on a minimized or maximized form. This error has the following causes and solutions:

- You tried to use the **Move** method or change the **Left**, **Top**, **Height**, or **Width** of the form while it was maximized or minimized.

  Check the **WindowState** property of the form before using the **Move** method or prevent the user from maximizing or minimizing the form.
- The Resize event occurred while the form was maximized or minimized and code in it tried to change the **Left**, **Top**, **Height**, or **Width** of the form.

  Rewrite the code in the Resize event procedure to check the **WindowState** property of the form and exit the procedure if **WindowState** is 1 - Minimized or 2 - Maximized.

Must specify index when using property array (Error 385)

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

You must specify an index value when referring to a property array. This error has the following cause and solution:

- You tried to refer to an element in a property array but omitted the index value specifying the element.

  Change your code to include the index value. For example, `List1.List` is invalid but `List1.List(3)` is valid.

'Item' property can't be set on this control (Error 387)

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

The property's setting can't be set at run time or can only be set under certain conditions. This error has the following causes and solutions:

- You tried to change the **Appearance**, **ControlBox**, **MinButton**, or **MaxButton** property settings for the form at run time.

  You may only read these property values at run time.
- You tried to set the **Visible** property to **False** for the last remaining visible submenu on a parent menu.

  You can't have a parent menu with no visible submenu items.

Can't set Visible property from a parent menu (Error 388)

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

The **Visible** property of a submenu item can't be set from its parent's menu code. This error has the following cause and solution:

- You tried to set the **Visible** property of a submenu item from a procedure associated with the submenu item's parent **Menu**.

  Place the code reference to the submenu item's **Visible** property in some other control's or object's procedure.

Invalid key (Error 389)

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

The key that was pressed is invalid.

'Item' property cannot be read at run time (Error 393)

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

The property is only available at design time. This error has the following cause and solution:

- You tried to read a property at run time that is only accessible at design time.
  Change your code and remove the reference to the property.

'Item' property is write-only (Error 394)

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

The property can’t be read. This error has the following cause and solution.

- A property can't be read; for example, `ctl.property = 3` might be legal, but "`Print ctl.property`" would generate this error.

  Change your code and remove the reference to the property.

Cannot use separator bar as menu name for this control (Error 395)

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

A **Menu** caption that has submenus can't be changed to a separator bar. This error has the following causes and solutions:

- At either design time or run time, you tried to change the **Caption** property on a parent or top-level **Menu** to a separator bar.

  Remove the dash character from the Caption text box in the Menu Editor dialog or the **Caption** property in the Properties window, or change the reference in your code.
- At run time, you tried to change the **Caption** property on a **Menu** which is checked, disabled, or has a shortcut key to a separator bar.

  Change the reference in your code or read the setting of the **Checked** or **Enabled** properties before trying to change the **Caption** property. You can't read the setting of the **Shortcut** property at run time.

'Item' property cannot be set within a page (Error 396)

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

Certain properties of the **Printer** object can't be changed within a page. This error has the following cause and solution:

- You tried to change the **Height**, **Width**, or **PaperSize** properties of the **Printer** object before advancing the print job to the next page.

  Use the **NewPage** method before changing the **Height**, **Width**, or **PaperSize** properties.

Can't load, unload, or set Visible property for top level menus while they are merged (Error 397)

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

The **Visible** property can't be set for a top-level menu while an ActiveX component's menu is merged with a container form's menu. This error has the following cause and solution:

- You tried to change the **Visible** property of a top level menu while a Visual Basic container form's menu and a linked or embedded ActiveX component's menu were sharing the same menu bar.

  Test to ensure that menus are not merged before setting **Visible** for top-level menus.

Form already displayed; can't show modally (Error 400)

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

You can't use the **Show** method to display a form as modal that is already visible. This error has the following cause and solution:

- You tried to use **Show**, with the *style* argument set to 1 - **vbModal**, on an already visible form.

  Use either the **Unload** statement or the **Hide** method on the form before trying to show it as a modal form.

Can't show non-modal form when modal form is displayed (Error 401)

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

When a modal form is displayed, you can't display another non-modal form. This error has the following cause and solution:

- You tried to use the **Show** method or set the **Visible** property to **True** on a non-modal form when another form is already displayed as modal.

  Use either the **Unload** statement or the **Hide** method on the modal form before attempting to use the **Show** method on a non-modal form.

Must close or hide topmost modal form first (Error 402)

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

The modal form you are trying to close or hide isn't on top of the z-order. This error has the following cause and solution:

- Another modal form is higher in the z-order than the modal form you tried to close or hide.

  First use either the **Unload** statement or the **Hide** method on any modal form higher in the z-order. A modal form is a form displayed by the **Show** method, with the *style* argument set to 1 - **vbModal**.

MDI forms cannot be shown modally (Error 403)

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

You may not set the modal option on an **MDIForm**. This error has the following cause and solution:

- You tried to use the **Show** method, with the *style* argument set to 1 - **vbModal**, on an MDI parent form.

  Remove the *style* argument reference from your code.
- You tried to compile an ActiveX DLL project that contains an **MDIForm**.

  MDFforms cannot be put in DLLs. Remove the **MDIForm** from the project and then re-compile, or change the Project Type from the Project Properties dialog box to Standard EXE and re-compile.

MDI child forms cannot be shown modally (Error 404)

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

You may not set the modal option on an MDI child form. This error has the following cause and solution:

- You tried to use the **Show** method, with the *style* argument set to 1 - **vbModal**, on an MDI child form.

  Remove the *style* argument reference from your code.

Permission to use object denied (Error 419)

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

You don't have the necessary permissions for the specified object. This error has the following cause and solution:

- You don't currently have the authority to access this object.
  To change your permission assignments, see your system administrator or the object's creator.

Only one MDI Form allowed (Error 426)

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

A project can only have one **MDIForm**. This error has the following causes and solutions:

- You tried to load a file containing an **MDIForm** into a project that already has an **MDIForm** in it. Remove the file containing the loaded **MDIForm** before loading another.
- You tried to load a second instance of an **MDIForm** created with a **Dim** or **Set** statement. You can only create one instance of an **MDIForm** in a project.

Invalid object type; Menu control required (Error 427)

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

The first argument for the **PopupMenu** method must be a **Menu** control. This error has the following cause and solution:

- You specified some object other than a **Menu** control as the *menuname* argument for the **PopupMenu** method.

  Use the name of a **Menu** control with at least one submenu as the *menuname* argument.

Popup menu must have at least one submenu (Error 428)

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

You have defined a pop-up menu with no submenus. This error has the following cause and solution:

- You specified a **Menu** control that doesn't have a submenu as the *menuname* argument for the **PopupMenu** method.

  Use the name of a **Menu** control with at least one submenu as the *menuname* argument.

License information for this component not found. You do not have an appropriate license to use this functionality in the design environment (Error 429)

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

You are not a licensed user of the ActiveX control. This error has the following cause and solution:

- You tried to place an ActiveX control on a form at design time or tried to add a form to a project with an ActiveX control on it, but the associated information in the registry could not be found.

  The information in the registry may have been deleted or become corrupted. Reinstall the ActiveX control or contact the control vendor.

Invalid Clipboard format (Error 460)

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

The specified **Clipboard** format is incompatible with the method being executed. This error has the following causes and solutions:

- You tried to use the clipboard's **GetText** or **SetText** method with a **Clipboard** format other than **vbCFText** or **vbCFLink**.

  Remove the invalid format and specify one of the two valid formats.
- You tried to use the clipboard's **GetData** or **SetData** method with a **Clipboard** format other than **vbCFBitmap**, **vbCFDIB**, or **vbCFMetafile**.

  Remove the invalid format and specify one of the three valid graphics formats.
- You tried to use the DataObject's **GetData** or **SetData** method with a clipboard format in the range reserved by Windows for registered formats (&HC000-&HFFFF), but that clipboard format has not been registered with Windows.

  Use a clipboard format recognized by Visual Basic, or a clipboard format which has been registered with Windows through the Windows RegisterClipboardFormat API function.

Specified format doesn't match format of data (Error 461)

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

The specified **Clipboard** format is incompatible with the method being executed. This error has the following causes and solutions:

- You tried to use the clipboard's **GetText** or **SetText** method with a **Clipboard** format other than **vbCFText** or **vbCFLink**.

  Before using these methods, use the **GetFormat** method to test if the current contents of the **Clipboard** matches the specified format.
- You tried to use the clipboard's **GetData** or **SetData** method with a **Clipboard** format other than **vbCFBitmap**, **vbCFDIB**, or **vbCFMetafile**.

  Before using these methods, use the **GetFormat** method to test if the current contents of the **Clipboard** matches the specified graphics format.
- You tried to use the DataObject's **GetData** or **SetData** method with a clipboard format inappropriate to the data, for instance, using **vbCFDIB** with a metafile.

  Before using these methods, use the **GetFormat** method to test if the current contents of the DataObject matches the specified data format.

Can't create AutoRedraw image (Error 480)

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

Visual Basic can't create a persistent bitmap for automatic redraw of the form or picture. This error has the following cause and solution:

- There isn't enough available memory for the **AutoRedraw** property to be set to **True**.

  Set the **AutoRedraw** property to **False** and perform your own redraw in the Paint event procedure or make the **PictureBox** control or **Form** object smaller and try the operation again with **AutoRedraw** set to **True**.

Invalid picture (Error 481)

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

An invalid graphics format was assigned to the **Picture** property. This error has the following cause and solution:

- You tried to assign a graphics format other than a bitmap, icon, or Windows metafile to the **Picture** property of a form or control.

  Ensure that the file you are trying to load into the **Picture** property is a valid graphics file supported by Visual Basic.

Printer error (Error 482)

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

There is some problem that prevents printing. This error has the following causes and solutions:

- You don't have a printer installed from the Windows Control Panel.
  Open the Control Panel, double-click the Printers icon, and choose Add Printer to install a printer.
- Your printer isn't online.
  Physically switch the printer online.
- Your printer is jammed or out of paper.
  Physically correct the problem.
- You tried to print a form to a printer that can accept only text.
  Switch to an installed printer that can print graphics.

Printer driver does not support specified property (Error 483)

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

The printer driver for the printer in use doesn't support this property of the **Printer** object. This error has the following cause and solution:

- The effect of the properties of the **Printer** object depends on the driver supplied by the printer manufacturer. Some property settings may have no effect, or several different property settings may all have the same effect. Settings outside the accepted range may or may not produce an error.

  For more information, see the manufacturer's documentation for the specific driver.

Problem getting printer information from the system. Make sure the printer is set up correctly. (Error 484)

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

There is some problem that prevents getting printer information from the system. This error has the following causes and solutions:

- You don't have a printer installed from the Windows Control Panel.
  Open the Control Panel, double-click the Printers icon, and choose Add Printer to install a printer.
- Your printer isn't online.
  Physically switch the printer online.
- Your printer is jammed or out of paper.
  Physically correct the problem.

Invalid picture type (Error 485)

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

The resource file picture format you tried to load doesn't match the specified property of the object. This error has the following causes and solutions:

- You tried to use the **LoadResPicture** method to load a bitmap resource as the **Icon** property of a **Form**.

  Change the property of **Form** to the **Picture** property or change the *format* argument of **LoadResPicture** to **vbResIcon**.
- You tried to use the **LoadResPicture** method to load a cursor resource as some property of an object or control other than the **MousePointer** property.

  Change the property reference to **MousePointer**.

Can't empty Clipboard (Error 520)

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

The **Clipboard** was opened but could not be emptied. This error has the following cause and solution:

- Another application is using the **Clipboard** and will not release it to your application.

  Set an error trap for this situation in your code and provide a message box with Retry and Cancel buttons to allow the user to try again after a short pause.

Can't open Clipboard (Error 521)

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

The **Clipboard** has already been opened by another application. This error has the following cause and solution:

- Another application is using the **Clipboard** and will not release it to your application.

  Set an error trap for this situation in your code and provide a message box with Retry and Cancel buttons to allow the user to try again after a short pause.

The data binding DLL, 'item', could not be loaded. (Error 523)

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

Unable to load specified data binding DLL. Make sure the DLL is installed on your system.

'item' (Error 524)

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

The specified error has occurred in the Automation object currently running.

Data Access Error (Error 525)

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

A Data Access Object error has occurred that was not specifically trapped.

The given bookmark was invalid (Error 527)

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

You tried to set a bookmark to an invalid string. This error has the following causes and solutions:

- You set the **Bookmark** property to a string that wasn't saved after reading a record's **Bookmark** property.

  Save the string before setting the **Bookmark** property to it.
- You set the **Bookmark** property to a string that was saved after reading a record's **Bookmark** property, but that record has since been deleted.

  Test that the record with the saved **Bookmark** property still exists before trying to move to it.

Could not lock the database (Error 536)

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

The database file could not be locked. This error has the following causes and solutions:

- You tried to lock a table while opening it, but the table can't be locked because it is currently in use.
  Wait a moment, and then try the operation again.
- You tried to lock a record that is currently locked by another user.
  Wait for the other user to finish working with the record, and then try the operation again.

Could not lock the database (Error 541)

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

The database file could not be locked. This error has the following causes and solutions:

- You tried to lock a table while opening it, but the table can't be locked because it is currently in use.
  Wait a moment, and then try the operation again.
- You tried to lock a record that is currently locked by another user.
  Wait for the other user to finish working with the record, and then try the operation again.

The row has been deleted since the update was started (Error 542)

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

A record can't be updated because it was previously deleted. This error has the following cause and solution:

- Because the **LockEdits** property for the **Recordset** object was set to **False** (optimistic locking) while the record was being edited, another user was able to delete the record before it could be updated.

  Set the **LockEdits** property to **True** (pessimistic locking).

Client Site not available (Error 398)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgClientSiteNotAvailableError398C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgClientSiteNotAvailableError398S"}

The User Control or User Document cannot retrieve information from its container. This error has the following cause and solution:

- The User Control or User Document tried to get information from the container's Ambient properties or its Extender object before the container attached to it. This can happen if you try to access these objects before your InitProperties or ReadProperties events get fired, such as in the Initialize event, or for some containers in the Terminate event.

  To solve this problem, wait for an InitProperties or ReadProperties event before accessing these objects.

You can't put a Default or Cancel button on a User Control unless its DefaultCancel property is set (Error 399)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgYouCantPutDefaultOrCancelButtonOnUserControlUnlessItsDefaultCancelPropertyIsSetError399C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgYouCantPutDefaultOrCancelButtonOnUserControlUnlessItsDefaultCancelPropertyIsSetError399S"}

You can't set a control's **Default** or **Cancel** property to **True** on a User Control unless the User Control's **DefaultCancel** property is **True**. This error has the following causes and solutions:

- You tried to set the **Default** or **Cancel** property of a control contained by a User Control to **True**, but the **DefaultCancel** property of the User Control was set to **False**.

  Set the **DefaultCancel** property of the User Control to **True**. This allows the User Control to participate in form wide Cancel/Default behavior.

Form not found (Error 424)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgFormNotFoundError424C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgFormNotFoundError424S"}

The form was not found. This error has the following cause and solution:

- You tried to add a form to the **Forms** collection using the **Add** method, but there is no form class of that name. For example, `Forms.Add "Form2"`, where `Form2` doesn't exist.

  Make sure that the class name is available to your project.

Unable to bind to field: 'item' (Error 545)

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

The control was unable to bind to the field specified by the **DataField** property. This error has the following causes and solutions:

- The **DataField** property of the bound control was set to a field that is not in the record set provided by the **Data** control.

  Set the **DataField** property of the bound control to a valid field. At design time, you can get a list of available fields by selecting the **DataField** property drop-down list from the Properties window. At run time, use the **Fields** collection of the **Data** control's **Recordset** to get a list of fields.
- The **RecordSource** property of the **Data** control was changed and the **Data** control's **Refresh** method was executed, but the **DataField** property of an associated bound control was not updated.

  Set the **DataField** property of the bound control to a valid field. At design time, you can get a list of available fields by selecting the **DataField** property drop-down list from the Properties window. At run time, use the **Fields** collection of the **Data** control's **Recordset** to get a list of fields.
- The **Recordset** property of the **Data** control was set to a new **Recordset** and a bound control was set to a field that is not in the record set provided by the **Data** control.

  Set the **DataField** property of the bound control to a valid field. At design time, you can get a list of available fields by selecting the **DataField** property drop-down list from the Properties window. At run time, use the **Fields** collection of the **Data** control's **Recordset** to get a list of fields.

DataObject formats list may not be cleared or expanded outside of the OLEStartDrag event (Error 672)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgDataObjectFormatsListMayNotBeClearedOrExpandedOutsideOfOLEStartDragEventError672C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgDataObjectFormatsListMayNotBeClearedOrExpandedOutsideOfOLEStartDragEventError672S"}

The DataObject formats list may not be cleared or expanded outside of the OLEStartDrag event. This error has the following cause and solution:

- You tried to clear or expand the DataObject formats list with the **SetData** method in an event other than the OLEStartDrag event. This is not supported. For example, you may have placed the code `Data.SetData [value], format`, where *format* was not already supported by that data object, in some event other than the OLEStartDrag event.

  When expanding or clearing the DataObject formats list, make sure that you do it in the OLEStartDrag event.

Expected at least one argument (Error 673)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgExpectedAtLeastOneArgumentError673C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgExpectedAtLeastOneArgumentError673S"}

The method expected at least one argument. This error has the following cause and solution:

- The user tried to call the **DataObject.SetData** method without any parameters.

  Although both parameters are optional, you must call the **DataObject.SetData** method with at least one parameter.

Recursive invocation of OLE drag and drop (Error 674)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgIllegalRecursiveInvocationOfOLEDragDropError674C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgIllegalRecursiveInvocationOfOLEDragDropError674S"}

There was an attempt to recursively invoke an OLE drag and drop operation. This error has the following cause and solution:

- The user tried to invoke a new OLE drag and drop operation using the **OLEDrag** method while an OLE drag and drop operation was currently in process. This is not allowed.

  Don't use the **OLEDrag** method while an OLE drag and drop operation is in process.

Illegal parameter. Can't write arrays (Error 328)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgIllegalParameterCantWriteArraysError328C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgIllegalParameterCantWriteArraysError328S"}

An illegal parameter was passed to the method. This error has the following cause and solution:

- In the WriteProperties event of your User Control, you tried to do a **PropBag.WriteProperty** X, where X is an array. This isn't supported.

  You must write out each element of the array individually.

The ActiveX Designer's Type Information does not match what was saved. Unable to Load (Error 370)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgTheActiveXDesignersTypeInformationDoesNotMatchWhatWasSavedUnableToLoadError370C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgTheActiveXDesignersTypeInformationDoesNotMatchWhatWasSavedUnableToLoadError370S"}

An ActiveX Designer could not recreate the information that it saved, and won't be loaded. This error has the following cause and solution:

- The most common reasons for this to occur are – a forms package designer might not be able to load a control, a control might be missing a license, or a database designer might not be able to find ODBC.

  Re-install the ActiveX Designer to correctly register the designer, or contact the designer's vendor for more information.

You can't put a Default or Cancel button on a Property Page (Error 379)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgYouCantPutDefaultOrCancelButtonOnPropertyPageError379C"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgYouCantPutDefaultOrCancelButtonOnPropertyPageError379S"}

A Default or Cancel button cannot be put on a Property Page. This error has the following causes and solutions:

- You tried to set the **Default** or **Cancel** property of a control contained by a Property Page to **True**. This is not allowed.

  Property Pages already have Default/Cancel buttons on them and you're not allowed to override them.

The specified object can't be used as an owner form for Show() (Error 371)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgTheSpecifiedObjectCantBeUsedAsOwnerFormForShowError371C"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgTheSpecifiedObjectCantBeUsedAsOwnerFormForShowError371S"}

You specified an invalid object for the **OwnerForm** parameter of the **Show** method. This error has the following cause and solution:

- You specified the **OwnerForm** parameter for the **Show** method but didn't specify a valid owner.

  You must specify a valid Form, MDI Form, or ActiveX Designer for the **OwnerForm** parameter of the **Show** method.

Unable to show modal form within this context (Error 405)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgUnableToShowModalFormWithinThisContextError405C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgUnableToShowModalFormWithinThisContextError405S"}

You tried to show a modal form that would cause the current form to be unloaded. This is not allowed.

Can't load (or register) ActiveX control: 'item' (Error 367)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgCantLoadorRegisterCustomControlError367C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgCantLoadorRegisterCustomControlError367S"}

The specified ActiveX control cannot be loaded or registered. This error has the following cause and solution:

- You tried to load a form but one of the ActiveX controls on the form is not registered correctly, and its file cannot be located so it can't be auto-registered.

  This error occurs in compiled projects. Make sure that you distribute all necessary files when distributing your program to another user.

Control 'item' not found (Error 423)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgControlitemNotFoundError423C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgControlitemNotFoundError423S"}

Access to an invalid **VBControl** (or **ControlTemplate**) object. The control this object refers to may no longer exist or may be in an invalid state.

Top-level or invalid menu specified as PopupMenu default (Error 490)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgToplevelOrInvalidMenuSpecifiedAsPopupMenuDefaultError490C"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgToplevelOrInvalidMenuSpecifiedAsPopupMenuDefaultError490S"}

You specified a default item in the **PopupMenu** method but that item was not a valid menu item.

The default item must be one of the items already listed in the menu.

Can't print form image to this type of printer (Error 486)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgCantPrintFormImageToThisTypeOfPrinterError486C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgCantPrintFormImageToThisTypeOfPrinterError486S"}

The form cannot print to this type of printer. This error has the following cause:

- Your printer doesn't support RASTER capabilities or failed a StartPage during the printing of a form.

Data value named 'item' not found (Error 327)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgDataValueNameditemNotFoundError327C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgDataValueNameditemNotFoundError327S"}

The specified data value could not be found. This error has the following cause and solution:

- In the ReadProperties event of a **UserDocument** or **UserControl**, you passed in the name of a property to **PropBag.ReadProperty** that doesn't exist and you didn't specify a default value.

  To avoid this error, verify that the specified property exists or use the optional default parameter for the **PropBag.ReadProperty**.

Operation not valid in an ActiveX DLL (Error 369)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgOperationNotValidInDLLError369C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgOperationNotValidInDLLError369S"}

Some operations are not valid in an ActiveX DLL or ActiveX Control project. This error has the following causes:

- You set the **LinkMode** property of a form or control.
- You set or accessed the **App.TaskVisible** property.
- You set or accessed the **App.OleRequestPendingTimeout** property.
- You set or accessed the **App.OleServerBusyRaiseError** property.
- You set or accessed the **App.OleServerBusyTimeout** property.
- You set or accessed the **App.OleServerBusyMsgTitle** property.
- You set or accessed the **App.OleServerBusyMsgText** property.
- You set or accessed the **App.OleRequestPendingMsgTitle** property.
- You set or accessed the **App.OleRequestPendingMsgText** property.

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

Invalid object use (Error 425)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgInvalidObjectUseError425C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgInvalidObjectUseError425S"}

This error has the following causes and solution:

- You called the **SetData** or **Clear** methods of **DataObject** during an OLE drag-and-drop target event (OLEDragOver or OLEDragDrop).
- You called the **GetData** method of **DataObject** during an OLE drag-and-drop source event (OLEStartDrag, OLECompleteDrag, OLEGiveFeedback, OLESetData).
- You tried to reference a **DataObject** object outside of the event in which it was passed in.
- You tried to use the **PropertyBag** object outside of the ReadProperties and WriteProperties events of a User Control.

In general, this error is caused by an illegal reference to an object. To solve the problem, remove the reference to that object or try referencing it in a different part of your program.

PropertyName parameter conflicts with the PropertyName of an AsyncRead in progress (Error 689)

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

You tried to execute the **AsyncRead** method with the same **PropertyName** of an **AsyncRead** that is currently in progress. This error has the following cause:

- For example, if you do the following in the UserControl event:

  ```
  AsyncRead "file:c:\winnt\winnt.bmp", vbAsyncTypePicture, "MyPicture"
  AsyncRead "file:c:\winnt\winnt.bmp", vbAsyncTypePicture, "MyPicture"
  ```

  You would get the error on the second **AsyncRead** method.

Can't find or load the required file urlmon.dll (Error 690)

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

This error occurs if you try to call the **AsyncRead** method and don't have the urlmon.dll file. The Setup Wizard distributes the urlmon.dll file when you distribute your application.

Verify that this file is available in the Windows SYSTEM or SYSTEM32 directory.

An unknown protocol was specified in Target parameter (Error 693)

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

You must specify a valid protocol in the target parameter for the **AsyncRead** method. This error has the following cause and solution:

- You specified an invalid protocol in the target parameter for the **AsyncRead** method.

From a UserControl or UserDocument, you must specify a known value such as HTTP: or FILE: as the prefix for the **Target** parameter of the **AsyncRead** method. For example, "h`ttp://vbdocs/userdoc1.vbd"` or `"file:\\vbdocs\userdoc1.vbd"`.

Object server 'item' not correctly registered or not found (Error 339)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgObjectServeritemNotCorrectlyRegisteredOrNotFoundError339C"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgObjectServeritemNotCorrectlyRegisteredOrNotFoundError339S"}

This error has the following causes and solution:

- The specified file was moved or deleted.
- The specified file was incorrectly registered.

Reinstall the specified file to correctly register it. See the object's vendor for more information.

'item' cannot be set while loading (Error 378)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgitemCannotBeSetWhileLoadingError378C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgitemCannotBeSetWhileLoadingError378S"}

You cannot set the specified value while loading. This error has the following cause and solution:

- You tried to modify a property of the **Extender** object from a UserControl, but the value cannot be changed while the container is loading. For example, you tried to change the **Extender.Name** property in your ReadProperties event.

  To solve this problem, move your code to an event that occurs later in the load process.

Non-modal forms cannot be displayed in this host application from an ActiveX DLL (Error 406)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgNonmodalFormsCannotBeDisplayedInThisHostApplicationFromActiveXDLLError406C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgNonmodalFormsCannotBeDisplayedInThisHostApplicationFromActiveXDLLError406S"}

Non-modal forms cannot be displayed from most host applications. This error has the following cause and solution:

- You have code in your ActiveX DLL that displays a non-modal form.

  If you need to display a modal form, change your project type to ActiveX EXE and rebuild your project.

Non-intrinsic OLE drag and drop formats used with SetData require Byte array data. GetData may return more bytes than were given to SetData (Error 675)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgNonintrinsicOLEDragDropFormatsUsedWithSetDataRequireByteArrayDataGetDataMayReturnMoreBytesThanWereGivenToSetDataError675C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgNonintrinsicOLEDragDropFormatsUsedWithSetDataRequireByteArrayDataGetDataMayReturnMoreBytesThanWereGivenToSetDataError675S"}

You specified a non-intrinsic data format in the **SetData** method but didn't pass the data in a byte array. Make sure to pass data to the **SetData** method in a byte array.

When retrieving the data, **GetData** may return more bytes (depending on the operating system) than were actually passed in with **SetData**. The target should expect this and interpret the data in a meaningful way.

Requested data was not supplied to the DataObject during the OLESetData event (Error 676)

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

The requested data was not supplied to the **DataObject** during the OLESetData event. This error has the following causes:

- The programmer specified a data format but did not specify the data itself during the OLEStartDrag event and didn't handle the OLESetData event.
- The programmer specified a data format but did not specify the data itself during the OLEStartDrag event and didn't provide the actual data during the OLESetData event (or else provided the data for the wrong format).

Failure in AsyncRead (Error 688)

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

A general error occurred during the **AsyncRead** method but no detailed information was returned.

One or more files in the project have changed. Do you wish to save the changes now?

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

One of the property settings or a code statement has changed for one or more of the files in the project and the changes have not yet been saved. This is your last chance to save the changes before loading another project.

Control must be same type as rest of array

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

You attempted to give a control the same **Name** property setting as an existing control array of a different type. All controls in a control array must be the same type.

Can't clear Index property without changing Name

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

This control is part of a control array. If you want to delete its index value, which deletes it from the control array, you must first rename it with a unique **Name** property setting.

Item' is a control name

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

A control already has this name. A form can't have the same name as a control in the same project.

Menu item skipped a level

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

Menu levels must be contiguous. Menus can be cascaded to five levels. An item at level 2 can open a pop-up menu at level 3, and an item at level 3 can open a pop-up menu at level 4.

In the Menu Editor dialog, make sure menu items are indented only one level beyond the previous level.

Menu control must have a name

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

A menu item has been created without assigning a name to it. Unlike other controls in Visual Basic, a **Menu** control requires you to assign names as the control is created, because Visual Basic doesn't supply a default name for menu items.

Menu control array element must have an index

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

An item has been added to a **Menu** control array without assigning an index value to the item. Unlike other control arrays in Visual Basic, **Menu** control arrays require you to assign indices as the item is created, because Visual Basic doesn't supply a default **Index** property setting for **Menu** control array items.

Menu control array indexes must be in ascending order

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

An index value assigned to an item in a **Menu** control array is out of order with the other items. The indices of a **Menu** control array don't have to be contiguous (for example, they could be 1, 3, and 5), but they must be in ascending order according to their location on the menu. Get the next available index number before assigning a number to the index.

Menu control array elements must be contiguous and within the same submenu

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

You have tried to add an item to a **Menu** control array in which the other items in the array are adjacent to the current item. The indices of the items in a **Menu** control array don't have to be contiguous (for example, 2, 3, 4), but they must be in ascending order according to their location on the menu. The items in a **Menu** control array, however, must be located next to each other within the same menu or pop-up menu.

AccessKeyPress Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtAccessKeyPressEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtAccessKeyPressEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtAccessKeyPressEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtAccessKeyPressEventS"}

Occurs when the user of the control presses one of the control's access keys, or when the Enter key is pressed when the developer has set the **Default** property to **True**, or when the Escape key is pressed when the developer has set the **Cancel** property to **True**. The **Default** property and the **Cancel** property are enabled by the author of the control setting the **DefaultCancel** property to **True**.

Syntax

Sub *object*_**AccessKeyPress(***KeyAscii* **As Integer)**

The **AccessKeyPress** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.
KeyAscii	An integer that contains the Ascii value of the key (without the ALT) that caused the **AccessKeyPress** event to fire, in the same manner as the standard KeyPress event.

AccessKeys Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAccessKeysPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAccessKeysPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAccessKeysPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAccessKeysPropertyS"}

Returns or sets a string that contains the keys that will act as the access keys (or hot keys) for the control.

Syntax

object.**AccessKeys** [= *AccessKeyString*]

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

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.
AccessKeyString	A string containing the keys that will act as the access keys.

Remarks

The **AccessKeys** property is a string that contains all the access keys for the control. As an example, to set the letters S and Y as the access keys, the **AccessKeys** property would be set to “sy”.

When a user presses one of the access keys in conjunction with the ALT key, the control will get the focus (depending on the setting of the **ForwardFocus** property).

Access keys for constituent controls are implicitly included as AccessKeys, although they will not appear in the **AccessKeys** property.

Alignable Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAlignablePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAlignablePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAlignablePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAlignablePropertyS"}

Returns or sets a value determining if a control is alignable, and can use the extender **Align** property. The **Alignable** property is read/write at the control's authoring time, and not available at the control's run time.

Settings

The settings for **Alignable** are:

Setting	Description
True	The control is alignable; the container will add the **Align** property to the extender object.
False	The control is not alignable. This is the default value.

Remarks

The alignment of the control itself will be handled by the container; the author of the control can use the **Align** extender property to decide how to redraw the control and arrange the constituent controls in response to an alignment.

Not all containers support alignable controls. Error trapping should be used if you access the **Align** extender property to determine how your control has been aligned.

AmbientProperties Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjAmbientObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjAmbientObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjAmbientObjectP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjAmbientObjectM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjAmbientObjectE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjAmbientObjectS"}

An **AmbientProperties** object holds ambient information from a container to suggest behavior to controls contained within the container.

Remarks

Containers provide ambient properties in order to suggest behavior to controls. As an example, **BackColor** is one of the standard ambient properties; the container is suggesting what the well behaved control should set its back color property to.

The **AmbientProperties** object's properties are the ambient properties of the container. These properties are read-only.

Some ambient properties are standard, while others are specific to certain containers. A control may access non-standard ambient properties, but this will make the control container-specific. The control should handle the case where an ambient property is not present in the current container.

When the control is compiled, Visual Basic has no way of knowing what container-specific ambient properties may be available when the control is run; therefore references to container-specific ambient properties will always be late bound.

The **AmbientProperties** object is not available when the **Initialize** event is raised; but is available when the **InitProperties** event or **ReadProperties** event is raised.

The **AmbientProperties** object has several standard properties:

The BackColor property, a Color that contains the suggested interior color of the contained control. The Visual Basic supplied default if the container does not support this property is 0x80000005: the system color for a window background.

The DisplayAsDefault property, a Boolean that specifies if the control is the default control. The Visual Basic supplied default if the container does not support this property is **False**.

The DisplayName property, a String containing the name that the control should display for itself. The Visual Basic supplied default if the container does not support this property is an empty string: "".

The Font property, a Font object that contains the suggested font information of the contained control. The Visual Basic supplied default if the container does not support this property is MS Sans Serif 8.

The ForeColor property, a Color that contains the suggested foreground color of the contained control. The Visual Basic supplied default if the container does not support this property is 0x80000008: the system color for window text.

The LocaleID property, a Long that specifies the language and country of the user. The Visual Basic supplied default if the container does not support this property is the current system locale ID.

The MessageReflect property, a Boolean that specifies if the container supports message reflection. The Visual Basic supplied default if the container does not support this property is **False**.

The Palette property, a Picture object who's palette specifies the suggested palette for the contained control.

The RightToLeft property, a Boolean that indicates the text display direction and control visual appearance on a bidirectional system. The Visual Basic supplied default if the container does not support this property is **False**.

The ScaleUnits property, a String containing the name of the coordinate units being used by the container. The Visual Basic supplied default if the container does not support this property is an empty string: “”.

The ShowGrabHandles property, a Boolean that specifies if the container handles the showing of grab handles. The Visual Basic supplied default if the container does not support this property is **True**.

The ShowHatchings property, a Boolean that specifies if the container handles the showing of hatching. The Visual Basic supplied default if the container does not support this property is **True**.

The SupportsMnemonics property, a Boolean that specifies if the container handles access keys for the control. The Visual Basic supplied default if the container does not support this property is **False**.

The TextAlign property, an enumeration that specifies how text is to be aligned. The Visual Basic supplied default if the container does not support this property is **0 - General Align**.

The UserMode property, a Boolean that specifies if the environment is in design mode or end user mode. The Visual Basic supplied default if the container does not support this property is **True**.

The UIDead property, a Boolean that specifies if the User Interface is nonresponsive. The Visual Basic supplied default if the container does not support this property is **False**.

Ambient Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAmbientPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAmbientPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAmbientPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAmbientPropertyS"}

Returns an AmbientProperties object holding the ambient properties of the container. The **Ambient** property is not available at the control's authoring time, and read-only at the control's run time.

Syntax

object.**Ambient**

The **Ambient** property syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

AmbientChanged Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtAmbientChangedEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtAmbientChangedEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtAmbientChangedEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtAmbientChangedEventS"}

Occurs when an ambient property's value changes.

Syntax

Sub *object*_**AmbientChanged(***PropertyName* **As String)**

The **AmbientChanged** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.
PropertyName	A string that identifies the ambient property that has changed.

Remarks

Using *PropertyName*, the control can access the **AmbientProperties** object in the **Ambient** property to check for the new value of the changed ambient property.

If an instance of the control is placed on a Visual Basic form, and the **FontTransparent** property of the form is changed, the **AmbientChanged** event will not be raised.

ApplyChanges Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtApplyChangesEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtApplyChangesEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtApplyChangesEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtApplyChangesEventS"}

Occurs when the user presses the OK button or the Apply button on the property page, or when property pages are switched by selecting tabs.

Syntax

Sub *object*_**ApplyChanges()**

The **ApplyChanges** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

When the **ApplyChanges** event is raised, the author of the property page needs to handle the setting of all the new property values to the controls; hopefully the author kept track of which properties were changed, otherwise all properties will have to be set. To know what controls are to be changed, use the **SelectedControls** property.

The **ApplyChanges** event will be raised only if the **Changed** property is set to **True**.

AsyncRead Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthAsyncReadC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthAsyncReadX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthAsyncReadA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthAsyncReadS"}

Start the reading in of data by the container from a file or URL asychronously.

Syntax

object.**AsyncRead** *Target*, *AsyncType* [, *PropertyName*]

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

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.
Target	A string expression specifying the location of the data. This can be a path or a URL.
AsyncType	An integer expression specifying how the data will be presented, as described in Settings.
PropertyName	An optional string expression specifying the name of the property to be loaded.

Settings

The settings for *AsyncType* are:

Setting	Description
vbAsyncFile	The data is provided in a file that is created by Visual Basic.
vbAsyncByte	The data is provided as a byte array that contains the retrieved data. It is assumed that the control author will know how to handle the data.
vbAsyncPicture	The data is provided in a Picture object.

Remarks

Once the data that is requested by the **AsyncRead** method is available the **AsyncReadComplete** event will be raised in the object. The asyncronous read may be canceled before it is completed by calling the **CancelAsyncRead** method.

The *PropertyName* parameter can be any arbitrary name, since it's only function is to act as an identifier for this particular data request. The value in *PropertyName* is used to identify the particular asychronous read to cancel in the **CancelAsyncRead** method, and the value in *PropertyName* is also used to identify the particular asynchronous read that has completed in the **AsyncReadComplete** event.

AsyncReadComplete Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtAsyncReadCompleteC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtAsyncReadCompleteX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtAsyncReadCompleteA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtAsyncReadCompleteS"}

Occurs when the container has completed an asychronous read request.

Syntax

Sub *object*_**AsyncReadComplete(***PropertyValue* **As AsyncProperty)**

The **AsyncReadComplete** event syntax has these parts:

Part	Description	
object	An object expression that evaluates to an object in the **Applies To** list.	
PropertyValue	An **AsyncProperty** object that contains the following properties:	
	Value	A Variant containing the results of the asychronous read. This is the default property.
	PropertyName	A string containing the property name that was passed in the **AsyncRead** method.
	AsyncType	An integer specifying the type of the data in the *Value* property, as described in Settings.

Settings

The settings for *AsyncType* are:

Setting	Description
vbAsyncFile	The *Value* property contains a string that is a path to a temporary file that contains the data.
vbAsyncByte	The *Value* property contains a byte array that contains the data.
vbAsyncPicture	The *Value* property contains a picture object of the correct format.

Remarks

The value in *PropertyName* specifies the particular asychronous data read request that has completed, and matches the value given in a previous **AsyncRead** method invocation.

Error handling code should be placed in the **AsyncReadComplete** event procedure, because an error condition may have stopped the download. If this was the case, that error will be raised when the the **Value** property of the **AsyncProperty** object is accessed.

BackStyle Property (UserControl Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBackStylePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBackStylePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBackStylePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBackStylePropertyS"}

Returns or sets a value indicating the type of the control's background. The **BackStyle** property is read/write at the control's authoring time, and read-only at the control's run time.

object.**BackStyle** [= *enum*]

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

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.
enum	An enumerated value that determines how the background of the control will be displayed, as described in Settings.

Settings

The settings for *enum* are:

Setting	Description
0-Transparent	Transparent background. Controls behind this control and the containing form's background will show through this control's blank areas. The area of the control's display can then be divided into two areas: the portion that is part of a constituent control, and the rest. This latter area of the control display area cannot be drawn on, and the portions of controls that are placed on the control by the developer or end user that fall into this latter area will also be invisible. Mouse events that fall in this latter area will not be given to the control, but rather to the underlying container.
1-Opaque	Opaque background. This is the default.
2-TransparentPaint	Transparent background. The difference between this option and **0-Transparent** is that controls behind this control and the containing form's background will show through this control's blank areas, but the entire area of this control can be drawn upon, controls placed on this control will not be invisible, and all mouse events that fall within this control will be given to this control.

BorderStyle Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBorderStylePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBorderStylePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBorderStylePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBorderStylePropertyS"}

Returns or sets a value indicating what the control's border style is. The **BorderStyle** property is read/write at the control's authoring time, and read-only at the control's run time.

object.**BorderStyle** [= *enum*]

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

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.
enum	An enumerated value that determines what the border style of the control is, as described in Settings.

Settings

The settings for *enum* are:

Setting	Description
0-None	No border. This is the default value.
1-Fixed Single	A single line is drawn around the control.

CancelAsyncRead Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthCancelAsyncReadC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthCancelAsyncReadX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthCancelAsyncReadA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthCancelAsyncReadS"}

Cancel an asychronous data request.

Syntax

object.**CancelAsyncRead** [*PropertyName*]

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

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.
PropertyName	An optional string expression specifying the name of the asychronous data request to cancel.

Remarks

Only the asychronous data read request specified by *PropertyName* is canceled; all others continue normally.

The value in *PropertyName* specifies the particular asychronous data read request to cancel, and should match the value given in a previous **AsyncRead** method invocation. If *PropertyName* is not given, then the last **AsyncRead** method invocation that did not give a *PropertyName* will be canceled.

CanGetFocus Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproCanGetFocusPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCanGetFocusPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCanGetFocusPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCanGetFocusPropertyS"}

Returns or sets a value determining if a control can receive focus. The **CanGetFocus** property is read/write at the control's authoring time, and not available at the control's run time.

Settings

The settings for **CanGetFocus** are:

Setting	Description
True	The control can receive focus. If the control contains constituent controls, the control itself will be unable to receive the focus unless none of its constituent controls can receive the focus. It is up to the author of the control to write the code that draws a focus rectangle on the control when it does receive focus. This is the default value.
False	The control cannot receive focus.

Remarks

As long as the control contains at least one constituent control that has been set to receive the focus, **CanGetFocus** cannot be set to **False**. If **CanGetFocus** is **False**, then no constituent control can be set to receive the focus.

CanPropertyChange Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthCanPropertyChangeMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthCanPropertyChangeMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthCanPropertyChangeMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthCanPropertyChangeMethodS"}

Asks the container if a property bound to a data source can have its value changed. The **CanPropertyChange** method is most useful if the property specified in *PropertyName* is bound to a data source.

Syntax

object.**CanPropertyChange** *PropertyName*

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

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.
PropertyName	A string expression that represents a name of the property that the control is requesting to change.

Return values

The possible return values for **CanPropertyChange** are:

Setting	Description
True	The property specified in *PropertyName* can be changed at this time.
False	The property specified in *PropertyName* cannot be changed at this time; the container has the bound data table open as read only. Do not set the property value; doing so may cause errors in some control containers.

Remarks

The control should always call **CanPropertyChange** before changing the value of a property that can be data-bound.

At present, **CanPropertyChange** always returns **True** in Visual Basic, even if the bound field is read-only in the data source. Visual Basic doesn't raise an error when the control attempts to change a read-only field; it just doesn't update the data source.

As an example, the following code shows how the **CanPropertyChange** method is used:

```
Public Property Let Address(ByVal cValue As String)
  If CanPropertyChange("Address") Then
     m_Address = cValue
     PropertyChanged "Address"
  End If
End Property
```

Changed Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproChangedPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproChangedPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproChangedPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproChangedPropertyS"}

Returns or sets a value indicating that a value of a property on a property page has changed. The **Changed** property is not available at the property page's authoring time, and read/write at the property page's run time.

Syntax

object.**Changed** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.
boolean	A boolean value that determines if a property on the property page has been changed, making the property page dirty.

Settings

The settings for *boolean* are:

Setting	Description
True	The property page is now dirty, since a the value of a property on the page has been changed.
False	The property page is not dirty, and no properties on the page have had their value changed.

Remarks

When the user changes the value of properties on a property page, these changes should not take effect immediately; instead, the changes should be applied only if the user presses the **Apply** button, the **OK** button, or changes property pages by selecting tabs. This allows the user to easily back out of any changes that have been made to a property page.

The **Changed** property should be set to **True**, for example, when a user changes a property value on a property page. Setting the **Changed** property to **True** would notify the property page to make available the **Apply** button.

ContainedControls Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproContainedControlsPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproContainedControlsPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproContainedControlsPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproContainedControlsPropertyS"}

Returns a collection of the controls that were added to the control by the developer or the end user at the control's run-time. The **ContainedControls** property is not available at the control's authoring time, and read-only at the control's run time.

Syntax

object.**ContainedControls**

The **ContainedControls** property syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

The **ContainedControls** collection is filled with all the controls that were added to the control by the developer or the end-user. The control can use the **ContainedControls** collection to perform operations on any of these contained controls.

This collection functions in a similar manner to the **Controls** collection on a form.

In order to allow contained controls to be placed on the control, the **ControlContainer** property must be **True**.

Contained controls cannot be added or removed through this **ContainedControls** collection; the contained controls must be changed in whatever manner the container allows.

The **ContainedControls** property may not be supported by all containers, even though the container may support the control having contained controls; Visual Basic forms do support this property. If this property is not supported, then calls to the **ContainedControls** collection will cause errors; use error handling when accessing the collection. Note, however, that if error handling is done while in an event procedure such as the **InitProperties** event procedure or the **ReadProperties** event procedure, the error handler should not raise an error event; doing this may be fatal to the container.

The **ContainedControls** collection is not available when the **Initialize** event is raised; but is available when the **InitProperties** event or **ReadProperties** event is raised.

Once the **ContainedControls** collection is present, it may not immediately contain references to the controls a developer has placed on the control. For example, if the control is on a Visual Basic form, the **Count** property of the **ContainedControls** collection will be zero until after the **ReadProperties** event procedure has executed.

ControlContainer Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproControlContainerPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproControlContainerPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproControlContainerPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproControlContainerPropertyS"}

Returns or sets a value determining if a control can contain controls placed on it by the developer or the end user at the control's run time; in the same way the PictureBox control can contain other controls. The **ControlContainer** property is read/write at the control's authoring time, and not available at the control's run time.

Settings

The settings for **ControlContainer** are:

Setting	Description
True	The control can contain controls placed on it. If an instance of this control is placed on a container that is not aware of ISimpleFrame, support of contained controls will be disabled. The control will continue to work correctly in all other ways, but developers or end users will be unable to place controls on an instance of this control.
False	The control cannot contain controls placed on it. This is the default value.

Remarks

Contained control support does work on a Visual Basic form.

Contained controls placed on a control with a transparent background are only visible where their location overlaps any constituent controls. Mouse events will be passed to the contained control only if they occur where the contained control is visible.

DataBindings Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjDataBindingsCollectionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjDataBindingsCollectionX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vamthItem;vbobjDataBindingsCollectionP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjDataBindingsCollectionM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjDataBindingsCollectionE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjDataBindingsCollectionS"}

The **DataBindings** collection is an extender property that collects the bindable properties that are available to the developer and end-user.

Remarks

All bindable properties appear in the **DataBindings** collection at end user run time. At developer design time (control run time), only properties marked “Show in DataBindings collection at design time” will appear when the **DataBindings** property is accessed in the Properties window.

DefaultCancel Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDefaultCancelPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDefaultlCancelPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDefaultCancelPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDefaultCancelPropertyS"}

Returns or sets a value determining if a control can act as a standard command button. The **DefaultCancel** property is read/write at the control's authoring time, and not available at the control's run time.

Settings

The settings for **DefaultCancel** are:

Setting	Description
True	The control can act as a default or cancel command button. The container will add the **Default** and **Cancel** properties to the extender object. The presence of the **Default** and **Cancel** properties allow the control to act as a standard command button. The control can then set these added extender properties.
False	The control cannot act as a default or cancel command button. No constituent control can have its **Default** or **Cancel** property set to **True**. This is the default value.

Remarks

Setting the **Default** property to **True** and also having a constituent control with its **Default** property set to **True** will cause the constituent control to be pressed when the Enter key is pressed, otherwise the control's **AccessKeyPress** event will be raised when the Enter key is pressed.

Setting the **Cancel** property to **True** and also having a constituent control with its **Cancel** property set to **True** will cause the constituent control to be pressed when the Escape key is pressed, otherwise the control's **AccessKeyPress** event will be raised when the Escape key is pressed.

Important The status of a default or cancel button can change at any time. Code must be placed in the control's **AmbientChanged** event procedure to detect changes in the **DisplayAsDefault** property, and the control's appearance adjusted accordingly.

DisplayAsDefault Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDisplayAsDefaultPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDisplayAsDefaultPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDisplayAsDefaultPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDisplayAsDefaultPropertyS"}

Returns a boolean value to determine if the control is the default button for the container, and therefore should display itself as the default control.

Syntax

object.**DisplayAsDefault**

The **DisplayAsDefault** property syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Settings

The possible boolean return values from the **DisplayAsDefault** property are:

Setting	Description
True	The control is the default button.
False	The control is not the default button. If the container does not implement this ambient property, this will be the default value.

Remarks

Only one control in a container may be the default control; the container of the control will determine which control is currently the default control and notify that control through the **DisplayAsDefault** ambient property. The notified control should draw itself to show it is the default control. All other controls will have their **DisplayAsDefault** ambient property value be **False**.

Only button type controls may be default controls.

DisplayName Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDisplayNamePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDisplayNamePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDisplayNamePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDisplayNamePropertyS"}

Returns a string value that contains the name the control should display to identify itself in error messages.

Syntax

object.**DisplayName**

The **DisplayName** property syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

This ambient property is the way the control finds out what the container (such as Visual Basic) is calling this instance of the control. This string should be used in error messages as the name for the instance of the control.

If the container does not implement this ambient property, the default value will be an empty string.

EditAtDesignTime Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vastmRaiseEvent;vbproEditAtDesignTimePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproEditAtDesignTimePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproEditAtDesignTimePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproEditAtDesignTimePropertyS"}

Returns or sets a value determining if a control can become active during the developer design time. The **EditAtDesignTime** property is read/write at the control's authoring time, and not available at the control's run time.

Settings

The settings for **EditAtDesignTime** are:

Setting	Description
True	Allows the control to become active at the developer design time. An **Edit** verb will appear on the control's context menu. When the developer who uses the control chooses **Edit**, the control will become active and behave as it does at end user run time.
False	The control cannot become active at developer design time. This is the default value.

Remarks

The control will only remain active while it is selected. When the developer selects another control, this control will no longer be active, even if the developer clicks back on this control. The developer must select **Edit** again from the context menu to make the control active.

When the control is activated in this fashion, the events of the UserControl object will occur, so that the control can operate normally, but the control will be unable to raise any events. The **RaiseEvent** method will simply be ignored; it will not cause an error.

EditProperty Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtEditPropertyEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtEditPropertyEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtEditPropertyEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtEditPropertyEventS"}

Occurs when a property page is opened because of the developer pressing the ellipsis button to display a particular property for editing.

Syntax

Sub *object*_**EditProperty(***PropertyName* **As String)**

The **EditProperty** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.
PropertyName	A string that identifies the property that is to be displayed and edited by the property page.

Remarks

This event happens when a property is assigned a property page via the Attributes dialog box. Assigning a property page through the Attributes dialog box means that the property is displayed in the property window with an ellipsis (…) next to it, and the developer can press the ellipsis button and the property page is automatically opened; the **EditProperty** event is then raised, so that the property page author can put the cursor on the correct field.

EnterFocus Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtEnterFocusEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtEnterFocusEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtEnterFocusEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtEnterFocusEventS"}

Occurs when focus enters the object. The object itself could be receiving focus, or a constituent control could be receiving focus.

Syntax

Sub *object*_**EnterFocus()**

The **EnterFocus** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

This event is useful if *object* needs to know that the focus is now inside of it.

The **EnterFocus** event is raised before any **GotFocus** event; the **GotFocus** event will only be raised in *object* or constituent control of *object* that actually got the focus.

EventsFrozen Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproEventsFrozenPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproEventsFrozenPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproEventsFrozenPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproEventsFrozenPropertyS"}

Returns a value indicating if the container is currently ignoring events being raised by the control. The **EventsFrozen** property is not available at the control's authoring time, and read-only at the control's run time.

Syntax

object.**EventsFrozen**

The **EventsFrozen** property syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

When the **EventsFrozen** property is **True**, the container is ignoring any events that are being raised by the control. If the control wants to raise an event that cannot be lost, it must queue them up until **EventsFrozen** is **False**.

ExitFocus Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtExitFocusC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtExitFocusX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtExitFocusA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtExitFocustS"}

Occurs when focus leaves the object. The object itself could be losing focus, or a constituent control could be losing focus.

Syntax

Sub *object*_**ExitFocus()**

The **ExitFocus** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

This event is useful if *object* needs to know that the focus is now leaving it.

The **ExitFocus** event is raised after any **LostFocus** event; the **LostFocus** event will only be raised in *object* or constituent control of *object* that actually loses the focus.

Extender Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjExtenderObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjExtenderObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjExtenderObjectP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjExtenderObjectM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjExtenderObjectE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjExtenderObjectS"}

An **Extender** object holds properties of the control that are actually controlled by the container of the control rather than by the control itself.

Remarks

Some properties of a control are provided by the container rather than the control; these are extender properties. Examples of extender properties are: Name, Tag and Left. The control still needs to know what the value of these extender properties are, and sometimes needs to be able to change an extender property; the Extender object gives the control access to these properties.

Some extender properties are standard, while others are specific to certain containers. A control may access non-standard extender properties, but this will make the control container-specific. If the control makes use of an extender property, the control should handle the case where the extender property is not supported by the current container.

When the control is compiled, Visual Basic has no way of knowing what extender properties may be available when the control is run; therefore references to extender properties will always be late bound.

The **Extender** object is not available when the **Initialize** event is raised; but is available when the **InitProperties** event or **ReadProperties** event is raised.

The **Extender** object has several standard properties:

The Name property, a read only String that contains the user-defined name of the control.

The Visible property, a read/write Boolean that specifies if the control is visible or not.

The Parent property, a read only object that represents the container of the control, such as a form in Visual Basic.

The Cancel property, a read only Boolean that indicates that the control is the default Cancel button for the container.

The Default property, a read only Boolean that indicates that the control is the default button for the container.

Visual Basic provides several more extender methods, properties and events; other containers are not guaranteed to provide these extender methods, properties and events. These Visual Basic specific extender methods, properties and events are:

The Container property, a read only object that represents the visual container of the control.

The DragIcon property, a read/write Picture that specifies the icon to use when the control is dragged.

The DragMode property, a read/write Integer that specifies if the control will automatically drag, or if the user of the control must call the **Drag** method.

The Enabled property, a read only Boolean that specifies if the control is enabled. This extender property is not present unless the control also has an Enabled property with the correct procedure ID. For additional information, refer to the topic “Allowing Your Controls to be Enabled and Disabled” in Chapter 9: Building ActiveX Controls.

The Height property, a read/write Integer that specifies the height of the control in the container’s scale units.

The HelpContextID property, a read/write Integer that specifies the context ID to use when the F1 key is pressed when the control has the focus.

The Index property, a read only Integer that specifies the position in a control array this instance of the control occupies.

The Left property, a read/write Integer that specifies the position of the left edge of the control to the left edge of the container, specified in the container's scale units.

The TabIndex property, a read/write Integer that specifies the position of the control in the tab order of the controls in the container.

The TabStop property, a read/write Boolean that specifies if Tab will stop on the control.

The Tag property, a read/write String that contains a user-defined value.

The ToolTipText property, a read/write String that contains the text to be displayed when the cursor hovers over the control for more than a second.

The Top property, a read/write Integer that specifies the position of the top edge of the control to the top edge of the container, specified in the container's scale units.

The WhatThisHelpID property, a read/write Integer that specifies the context ID to use when the What's This pop-up is used on the control.

The Width property, a read/write Integer that specifies the width of the control in the container's scale units.

The Drag method, a method to begin, end, or cancel a drag operation of the control.

The Move method, a method to move the position of the control.

The SetFocus method, a method to set the focus to the control.

The ShowWhatsThis method, a method to display a selected topic in a Help file using the What's This popup provided by Help.

The ZOrder method, a method to place the control at the front or back of the z-order within its graphical level.

The DragDrop event, an event that is raised when another control on the form is dropped on this control.

The DragOver event, an event that is raised when another control on the form is dragged over this control.

The GotFocus event, an event that is raised when this control gets the focus.

The LostFocus event, an event that is raised when this control loses the focus.

Extender Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproExtenderPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproExtenderPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproExtenderPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproExtenderPropertyS"}

Returns the Extender object for this control that holds the properties of the control that are kept track of by the container. The **Extender** property is not available at the control's authoring time, and read-only at the control's run time.

Syntax

object.**Extender**

The **Extender** property syntax has this part:

Part	**Description**
object	An object expression that evaluates to an object in the **Applies To** list.

ForwardFocus Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproForwardFocusPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproForwardFocusPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproForwardFocusPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproForwardFocusPropertyS"}

Returns or sets a value determining which control receives focus when one of the access keys for the control is pressed. The **ForwardFocus** property is read/write at the control's authoring time, and not available at the control's run time.

Settings

The settings for **ForwardFocus** are:

Setting	Description
True	The next control in tab order will receive focus when one of the access keys for the control is pressed.
False	If the **CanGetFocus** property is true, the control itself will receive focus when one of the access keys for the control is pressed. This is the default value.

Remarks

The **ForwardFocus** property allows the control to implement the behavior of a Label control that has an access key.

Access keys are set through the **AccessKeys** property. When an access key in conjunction with the ALT key is pressed, the control's **AccessKeyPress** event is raised.

GoBack Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthGoBackC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthGoBackX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthGoBackA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthGoBackS"}

Execute a hyperlink jump back in the history list.

Syntax

object.**GoBack**

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

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

If the object is in a container that supports OLE hyperlinking, then the container will jump to the location that is back in the history list. If the object is in a container that does not support OLE hyperlinking, then this method will raise an error.

GoForward Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthGoForwardC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthGoForwardX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthGoForwardA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthGoForwardS"}

Execute a hyperlink jump forward in the history list.

Syntax

object.**GoForward**

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

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

If the object is in a container that supports OLE hyperlinking, then the container will jump to the location that is forward in the history list. If the object is in a container that does not support OLE hyperlinking, then this method will raise an error.

GotFocus Event (UserControl Object and UserDocument Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtGotFocusEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtGotFocusEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtGotFocusEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtGotFocusEventS"}

Occurs in the object or constituent control when focus enters it.

Syntax

Sub *object*_**GotFocus()**

The **GotFocus** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

This **GotFocus** event is not the same **GotFocus** extender event that the developer who uses *object* handles. This **GotFocus** event is for the author of *object*, and is internal to object.

This event is useful if *object* needs to know that the focus is now on it.

Object itself can get focus only when the **CanGetFocus** property is **True** and there are no constituent controls that can receive the focus.

The **EnterFocus** event is raised before the **GotFocus** event.

Do not raise the **GotFocus** extender event from this event.

Hide Event (UserControl Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtHideEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtHideEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtHideEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtHideEventS"}

Occurs when the object's **Visible** property changes to **False**.

Syntax

Sub *object*_**Hide()**

The **Hide** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

In order to draw to the screen in Windows, any object must have a window, temporarily or permanently; Visual Basic ActiveX controls have permanent windows. Before a control has been sited on a form, its window is not on the container. The control receives **Hide** events when the window is removed.

While the control's window is on the form, the object receives a **Hide** event when the control's **Visible** property changes to **False**.

The control does *not* receive **Hide** events if the form is hidden and then shown again, or if the form is minimized and then restored. The control's window remains on the form during these operations, and its **Visible** property doesn't change.

If the control is being shown in an internet browser, a **Hide** event occurs when the page is moved to the history list.

If the control is used with earlier versions of Visual Basic than 5.0, the control will not receive **Hide** events at design time. This is because earlier versions of Visual Basic did not put any visible windows on a form at design time.

Hide Event (UserDocument Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtHideEventUserDocumentC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtHideEventUserDocumentX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtHideEventUserDocumentA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtHideEventUserDocumentS"}

Occurs when the object's **Visible** property changes to **False**.

Syntax

Sub *object*_**Hide()**

The **Hide** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

In order to draw to the screen in Windows, any object must have a window, temporarily or permanently. Visual Basic ActiveX documents have permanent windows. The UserDocument object receives **Hide** events when the window is removed.

While *object*'s window is on the container, *object* receives a **Hide** event when *object*'s **Visible** property changes to **False**.

Object does *not* receive **Hide** events if the container is hidden and then shown again, or if the container is minimized and then restored. *Object*'s window remains on the container during these operations, and its **Visible** property doesn't change.

If *object* is being shown in an internet browser, a **Hide** event occurs when the page is moved to the history list by navigating off *object* to another document, or when Internet Explorer 3.0 is terminated while *object* is being viewed or is still within the cache of active documents. Use the event to destroy any global object references before navigating to another document.

If *object* is used with earlier versions of Visual Basic than 5.0, *object* will not receive **Hide** events at design time. This is because earlier versions of Visual Basic did not put any visible windows on a form at design time.

InitProperties Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtInitPropertiesEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtInitPropertiesEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtInitPropertiesEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtInitPropertiesEventS"}

Occurs when a new instance of an object is created.

Syntax

Sub *object*_**InitProperties()**

The **InitProperties** event syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

This event allows the author of the object to initialize a new instance of the object. This event occurs only when a new instance of an object is being created; this is to allow the author of the object to distinguish between creating a new instance of the object and loading an old instance of the object.

By putting in code to initialize new instances in the **InitProperties** event rather than the **Initialize** event, the author can avoid cases where loading data through a **ReadProperties** event into an old instance of the object will undo the initialization of the object.

InvisibleAtRuntime Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproInvisibleAtRuntimePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproInvisibleAtRuntimePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproInvisibleAtRuntimePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproInvisibleAtRuntimePropertyS"}

Returns or sets a value determining if a control should not have a visible window at run time. The **InvisibleAtRuntime** property is read/write at the control's authoring time, and not available at the control's run time.

Settings

The settings for **InvisibleAtRuntime** are:

Setting	Description
True	Allows the control to not have a visible window at run time. The container of the control may keep the control invisible during run time, like the Timer control. The control is still active, and therefore the developer who uses the control can still write programs that can interact with the control. There will be no **Visible** property in the extender object.
False	The control acts as a normal control at run time, where the state of the **Visible** extender property determines the visibility of the control. This is the default value.

Remarks

Important Don't use the **Visible** extender property to make the control invisible at run time. If you do, the control will still have all the overhead of a visible control at run time. Furthermore, the extender properties are available to the developer and end user, who may make the control visible.

Some containers may not support the **InvisibleAtRuntime** property; in this case the control will be visible at run time.

Before creating a control that is invisible at run time, consider creating an ordinary object provided by an in-process code component (ActiveX DLL) instead. Objects provided by in-process code components require fewer resources than controls, even invisible controls. The only reason to implement an invisible control is to take advantage of a feature that is only available to ActiveX controls.

LocaleID Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproLocaleIDPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLocaleIDPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLocaleIDPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLocaleIDPropertyS"}

Returns a long value that contains the Locale identification (language and country) of the user.

Syntax

object.**LocaleID**

The **LocaleID** property syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

The **LocaleID** ambient property contains identification of the language and country of the current user. Using this identification, the control can modify its behavior and appearance to fit the language and country. This could be as simple as having error notifications in the language of the user, to more complex modifications of property, method, and event names in the language of the user.

If the container does not implement this ambient property, the default value will be the current System LocaleID.

LostFocus Event (UserControl Object and UserDocument Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtLostFocusEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtLostFocusEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtLostFocusEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtLostFocusEventS"}

Occurs in the object or constituent control when focus leaves it.

Syntax

Sub *object*_**LostFocus()**

The **LostFocus** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

This **LostFocus** event is not the same **LostFocus** extender event that the developer who uses *object* handles. This **LostFocus** event is for the author of *object*, and is internal to *object*.

This event is useful if *object* needs to know that the focus is now on it.

Object itself can get focus only when the **CanGetFocus** property is **True** and there are no constituent controls that can receive focus.

The **LostFocus** event is raised before the **ExitFocus** event.

MessageReflect Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproMesageReflectPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMesageReflectPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMesageReflectPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMesageReflectPropertyS"}

Returns a boolean value stating whether the control container handles message reflection automatically.

Syntax

object.**MessageReflect**

The **MessageReflect** property syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Settings

The possible boolean return values from the **MessageReflect** property are:

Setting	Description
True	The container for the control will reflect messages.
False	The container for the control cannot reflect messages. If the container does not implement this ambient property, this will be the default value.

Remarks

When a control is subclassed, there are certain messages that are normally sent to the parent control. Under normal conditions, these messages are actually reflected back to the sending control, so that the control can handle its own message. This message reflection can be handled by the container, which will reflect the messages back as events. The **MessageReflect** property tells if the container for the control does message reflection.

If the control is ever placed in a container that does not reflect messages, the operation of the control will be severely compromised; much of the operation of a control depends on reflected messages.

NavigateTo Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthNavigateToMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthNavigateToMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthNavigateToMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthNavigateToMethodS"}

Execute a hyperlink jump to the specified target.

Syntax

object.**NavigateTo** *Target* [**,** *Location* [, *FrameName*]]

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

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.
Target	A string expression specifying the location to jump to. This can be a document or a URL.
Location	A string expression specifying the location within the URL specified in *Target* to jump to. If *Location* is not specified, the default document will be jumped to.
FrameName	A string expression specifying the frame within the URL specified in *Target* to jump to. If *FrameName* is not specified, the default frame will be jumped to.

Remarks

If the object is in a container that supports OLE hyperlinking, then the container will jump to the specified location. If the object is in a container that does not support OLE hyperlinking, then an application that is registered as supporting hyperlinking is started to handle the request.

If *Target* does not specify a valid location, an error is raised.

ParentControls Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproParentControlsPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproParentControlsPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproParentControlsPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproParentControlsPropertyS"}

Returns a collection of the other controls in the control's container. The **ParentControls** property is not available at the control's authoring time, and read-only at the control's run time.

Syntax

object.**ParentControls**

The **ParentControls** property syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

In most cases, the container of the control will be a form; this collection functions in a similar manner to the **Controls** collection on the form, but will also contain the form itself.

This collection is useful if the control wants to perform some action on the controls on the form; the control can iterate through the collection.

Controls cannot be added or removed by the developer who uses the control through this collection; the controls must be changed in whatever manner the container allows.

The contents of this collection is determined entirely by the container.

PropertyBag Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjPropertyBagObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjPropertyBagObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjPropertyBagObjectP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjPropertyBagObjectM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjPropertyBagObjectE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjPropertyBagObjectS"}

A **PropertyBag** object holds information that is to be saved and restored across invocations of an object.

Remarks

A **PropertyBag** object is passed into an object through the **ReadProperties** event and the **WriteProperties** event in order to save and restore the state of the object. Using the methods of the **PropertyBag** object, the object can read or write properties of itself. The **ReadProperty** method of the **PropertyBag** object is used to read a value for a property, while the **WriteProperty** method of the **PropertyBag** object is used to write a value of a property. The value of a property can itself be an object; in that case the **PropertyBag** object will attempt to save it.

PropertyChanged Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthPropertyChangedMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthPropertyChangedMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthPropertyChangedMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthPropertyChangedMethodS"}

Notifies the container that a property's value has been changed.

Syntax

object.**PropertyChanged** *PropertyName*

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

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.
PropertyName	A string expression that represents a name of the property that the control has changed the value of.

Remarks

By notifying the container that a property's value has changed, the container can synchronize its property window with the new values of the object's properties. Also, the container would not know if an instance of the object needed to be saved (through raising a **WriteProperties** event) unless the container was notified that a property's value had changed.

This method needs to be called, for example, when a user changes a property value on a property page, or the object itself changes a property value. This method should also be called when a databound property is modified; otherwise the data source will not be updated.

Properties that are available only at run time do not need to call the PropertyChanged method, unless they can be data-bound.

As an example, the following code shows how the **PropertyChanged** method is used:

```
Public Property Let Address(ByVal cValue As String)
  m_Address = cValue
  PropertyChanged "Address"
End Property
```

PropertyPages Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproPropertyPagesPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPropertyPagesPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPropertyPagesPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPropertyPagesPropertyS"}

Returns or sets a string that is the name of a property page that is associated with a control.

Syntax

object.**PropertyPages(***index***)** [= *PropPageName*]

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

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.
index	Index into the string array.
PropPageName	A string containing the name of a property page in the project.

Remarks

PropertyPages property is a string array containing the names of the property pages in the project that are associated with this control. A property page may be added to the array by setting the last item in the array (which is always empty). A property page may be deleted from the array by setting that element in the array to an empty string.

The order of the names of property pages in the array determine the order in which pages appear in the property page's dialog box for the control.

Public Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproPublicPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPublicPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPublicPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPublicPropertyS"}

Returns or sets a value determining if a control can be shared with other applications. The **Public** property is read/write at the control's authoring time, and not available at the control's run time.

Settings

The settings for **Public** are:

Setting	Description
True	The control can be shared with other applications. This is the default for ActiveX Control project types.
False	The control cannot be shared with other applications. When the control is contained in an ActiveX Control project, the control cannot be seen outside of the ActiveX Control project. This means that other controls or other forms in the project can use the control, but outside applications cannot. This is the only valid value for project types other than ActiveX Control.

ReadProperties Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtReadPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtReadPropertiesX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtReadPropertiesA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtReadPropertiesS"}

Occurs when loading an old instance of an object that has a saved state.

Syntax

Sub *object*_**ReadProperties(***pb* **As PropertyBag)**

The **ReadProperties** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.
pb	An object of the type **PropertyBag** class that contains the saved data to load.

Remarks

When this event occurs, the object author can load in the saved state from *pb*, by calling the **ReadProperty** method of the **PropertyBag** object for each value that is to be loaded. This event occurs after the **Initialize** event.

Always include error trapping when handling the **ReadProperties** event, to protect the control from invalid property values that may have been entered by users editing the file containing the saved data with text editors. However, you should not raise an error in an event, since doing so may be fatal to the container, so any error trapping in the **ReadProperties** event procedure should not include raising errors.

ReadProperty Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthReadPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthReadPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthReadPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthReadPropertyS"}

Returns a saved value from a **PropertyBag** class object.

Syntax

object.**ReadProperty(***DataName*[**,** *DefaultValue*]**)**

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

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.
DataName	A string expression that represents a data value in the property bag.
DefaultValue	The value to be returned if no value is present in the property bag.

Remarks

The **ReadProperty** method will return the value of the saved data that is represented by the string expression *DataName*, or *DefaultValue* if there is no saved value. *DataName* should match the string expression that was used to store the saved data value in the property bag.

Note: Specifying a default value reduces the size of the file belonging to the container of the control. A line for the property is written to the file only if the value to be written is different from the default. Wherever possible, you should specify default values for the properties of the control when initializing, saving, and retrieving property values.

RightToLeft Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproLeftToRightPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLeftToRightPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLeftToRightPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLeftToRightPropertyS"}

Returns a boolean value that indicates the text display direction and controls the visual appearance on a bidirectional system.

Syntax

object.**RightToLeft**

The **RightToLeft** property syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Settings

The possible boolean return values from the **RightToLeft** property are:

Setting	Description
True	The control is running on a bi-directional platform, such as Arabic Windows95 or Hebrew Windows95, and text is running from right to left. The control should modify its behavior, such as putting vertical scroll bars at the left side of a text or list box box, putting labels to the right of text boxes, etc.
False	The control should act as though it was running on a non-bidirectional platform, such as English Windows95, and text is running from left to right. If the container does not implement this ambient property, this will be the default value.

ScaleUnits Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproScaleUnitsPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproScaleUnitsPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproScaleUnitsPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproScaleUnitsPropertyS"}

Returns a string value that is the name of the coordinate units being used by the container.

Syntax

object.**ScaleUnits**

The **ScaleUnits** property syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

This string represents the coordinates used by the container of the control, such as “twips”. This string can be used by the control as a units indicator when displaying coordinate values.

If the container does not implement this ambient property, the default value will be an empty string.

SelectedControls Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproSelectedControlsPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSelectedControlsPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSelectedControlsPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSelectedControlsPropertyS"}

Returns a collection that contains all the currently selected controls on the form. The **SelectedControls** property is not available at the property page's authoring time, and read-only at the property page's run time.

Syntax

object.**SelectedControls**

The **SelectedControls** property syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

This collection is useful to a property page in determining which controls are currently selected, and therefore which controls might need properties changed. Some containers only allow one control to be selected at once; in that case **SelectedControls** will only contain one control. Other containers allow more than one control to be selected at once; in that case there may be more than one control selected, and the property page must iterate through the controls in the **SelectedControls** collection and attempt to set the changed properties. Suitable error handling should be written to take care of the cases when a particular control in the collection does not have the changed property, or when the control raises an error when the property is set.

SelectionChanged Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtSelectionChangedEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtSelectionChangedEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtSelectionChangedEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtSelectionChangedsEventS"}

Occurs when the selection of controls on the form has changed.

Syntax

Sub *object*_**SelectionChanged()**

The **SelectionChanged** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

The firing of this event notifies the property page that the selection of controls has changed, and therefore the display of current property values may need to be updated. The **SelectedControls** property should be read to find out the new set of selected controls.

The **SelectionChanged** event is also raised when the property page is first brought up for a control.

Show Event (UserControl Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtShowEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtShowEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtShowEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtShowEventS"}

Occurs when the object's **Visible** property changes to **True**.

Syntax

Sub *object*_**Show()**

The **Show** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

In order to draw to the screen in Windows, any object must have a window, temporarily or permanently. Visual Basic ActiveX controls have permanent windows. Before a control has been sited on a form, its window is not on the container. The control receives **Show** events when the window is added.

While the control's window is on the form, the control receives a **Show** event when the control's **Visible** property changes to **True**.

The control does *not* receive **Show** events if the form is hidden and then shown again, or if the form is minimized and then restored. The control's window remains on the form during these operations, and its **Visible** property doesn't change.

If the control is being shown in an internet browser, a **Show** event occurs if the user returns to the page containing the control.

If the control is used with earlier versions of Visual Basic than 5.0, the control will not receive **Show** events at design time. This is because earlier versions of Visual Basic did not put any visible windows on a form at design time.

Show Event (UserDocument Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtShowEventUserDocumentC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtShowEventUserDocumentX"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtShowEventUserDocumentA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtShowEventUserDocumentS"}

Occurs when the object's **Visible** property changes to **True**.

Syntax

Sub *object*_**Show()**

The **Show** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Remarks

In order to draw to the screen in Windows, any object must have a window, temporarily or permanently; Visual Basic ActiveX documents have permanent windows. Before *object* has been sited on a form, its window is not on the container. The UserDocument object receives **Show** events when the window is added.

While *object*'s window is on the container, *object* receives a **Show** event when *object*'s **Visible** property changes to **True**.

Object does *not* receive **Show** events if the container is hidden and then shown again, or if the container is minimized and then restored. *Object*'s window remains on the container during these operations, and its **Visible** property doesn't change.

If *object* is being shown in an internet browser, a **Show** event occurs when the user navigates to the page.

If *object* is used with earlier versions of Visual Basic than 5.0, *object* will not receive **Show** events at design time. This is because earlier versions of Visual Basic did not put any visible windows on a form at design time.

ShowGrabHandles Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproShowGrabHandlesPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproShowGrabHandlesPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproShowGrabHandlesPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproShowGrabHandlesPropertyS"}

Returns a boolean value stating whether the control should show grab handles.

Syntax

object.**ShowGrabHandles**

The **ShowGrabHandles** property syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Settings

The possible boolean return values from the **ShowGrabHandles** property are:

Setting	Description
True	The control should show grab handles, if needed. If the container does not implement this ambient property, this will be the default value.
False	The control should not show grab handles.

Remarks

The default behavior for a control is to automatically show grab handles when the control is in a container that is in design mode (the control's run mode.) However, many containers do not want the control to show grab handles, preferring to handle the indication of control sizing in another way. The **ShowGrabHandles** property is how the container notifies the control of who is to display the sizing indications.

Note that all known containers prefer to handle the indication of control sizing themselves, and therefore set the **ShowGrabHandles** property to **False**. It is probably not necessary to actually handle the case when **ShowGrabHandles** is **True**.

ShowHatching Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproShowHatchingPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproShowHatchingPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproShowHatchingPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproShowHatchingPropertyS"}

Returns a boolean value stating whether the control should show hatching around the control.

Syntax

object.**ShowHatching**

The **ShowHatching** property syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Settings

The possible boolean return values from the **ShowHatching** property are:

Setting	Description
True	The control should show hatch marks, if needed. If the container does not implement this ambient property, this will be the default value.
False	The control should not show hatch marks.

Remarks

The default behavior for a control is to automatically show hatching when the control is in a container that is in design mode (the control's run mode) and the control is the one that has focus. However, many containers do not want the control to show hatching, preferring to handle the indication of control focus in another way. The **ShowHatching** property is how the container notifies the control of who is to display the control focus indications.

Note that Visual Basic forms do not implement this ambient property, and therefore the **ShowHatching** property is set to the default value of **True** when the control is placed in a Visual Basic form. However, Visual Basic does not expect the control to actually do anything in response to a **ShowHatching** value of **True**, therefore it is probably not necessary to actually handle the case when **ShowHatching** is **True**.

SupportsMnemonics Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproSupportsMnemonicsPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSupportsMnemonicsPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSupportsMnemonicsPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSupportsMnemonicsPropertyS"}

Returns a boolean value stating whether the control's container handles access keys for the control.

Syntax

object.**SupportsMnemonics**

The **SupportsMnemonics** property syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Settings

The possible boolean return values from the **SupportsMnemonics** property are:

Setting	Description
True	The container for the control does handle access keys.
False	The container for the control does not handle access keys. If the container does not implement this ambient property, this will be the default value.

Remarks

Most containers of controls are capable of handling all the processing of access keys for the controls contained within the container. This includes figuring out which control is to be given a particular access key. If a container is not capable of processing access keys, it is indicated with this **SupportsMnemonics** property, and the control can take action, such as not displaying the underlined character as an indication of keyboard accelerators.

TextAlign Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproTextAlignPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTextAlignPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTextAlignPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTextAlignPropertyS"}

Returns an enumerated value of type TextAlignChoices stating what kind of text alignment the container would like the control to do.

Syntax

object.**TextAlign**

The **TextAlign** property syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Settings

The possible enumerated return values from the **TextAlign** property are:

Setting	Description
0-General	General alignment: text to the left, numbers to the right. If the container does not implement this ambient property, this will be the default value.
1-Left	Align to the left.
2-Center	Align in the center.
3-Right	Align to the right.
4-FillJustify	Fill justify.

Remarks

This ambient property is the way that a container communicates to a contained control how to perform justification; this is a hint from the container that the control may or may not choose to follow.

ToolboxBitmap Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproToolboxBitmapPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproToolboxBitmapPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproToolboxBitmapPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproToolboxBitmapPropertyS"}

Returns or sets a bitmap that will be used as the picture representation of the control in the toolbox. The size of the space for the bitmap in the toolbox is 16x15 pixels; the bitmap specified by this property will be scaled to these dimensions if necessary. The **ToolboxBitmap** property is read/write at the control's authoring time, and not available at the control's run time.

Remarks

Important Do not assign an icon to the **ToolboxBitmap** property. Icons do not scale well to Toolbox bitmap size.

Visual Basic automatically uses the class name of the control as the tool tip text when users hover the mouse pointer over the icon in the Toolbox.

Tip When creating bitmaps, remember that for many forms of color-blindness, colors with the same overall level of brightness will appear to be the same. You can avoid this by restricting the bitmap to white, black, and shades of gray, or by careful color selection.

UIDead Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproUIDeadPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproUIDeadPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproUIDeadPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproUIDeadPropertyS"}

Returns a boolean value indicating whether the control should be responsive to the user or not.

Syntax

object.**UIDead**

The **UIDead** property syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Settings

The possible boolean return values from the **UIDead** property are:

Setting	Description
True	The control should not respond to the user.
False	The control should respond to the user. If the container does not implement this ambient property, this will be the default value.

Remarks

This property is typically used to indicate that the container is in break mode: during this mode, the control should not respond to any user input. That is, the control should ignore mouse clicks and keystrokes, and not change the mouse cursor even when the mouse is over the control window. A container such as a Visual Basic form would set this flag to TRUE when the programmer stops the program during execution—the container is not in design mode, yet not in run mode either; Visual Basic simply wants the controls to be inoperative.

UserMode Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproUserModePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproUserModePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproUserModePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproUserModePropertyS"}

Returns a boolean value indicating whether the control is being used by a form designer or a form user.

Syntax

object.**UserMode**

The **UserMode** property syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.

Settings

The possible boolean return values from the **UserMode** property are:

Setting	Description
True	The control is currently being used by a form user. If the container does not implement this ambient property, this will be the default value. In Visual Basic, this is Run Mode.
False	The control is currently being used by a form designer (the developer). In Visual Basic, this is Design Mode.

WriteProperties Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtWritePropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtWritePropertiesX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtWritePropertiesA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtWritePropertiesS"}

Occurs when an instance of an object is to be saved. This event signals to the object that the state of the object needs to be saved, so that the state can be restored later. In most cases, the state of the object consists only of property values.

Syntax

Sub *object*_**WriteProperties(***pb* **As PropertyBag)**

The **WriteProperties** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.
pb	An object of the type **PropertyBag** class to write the data to.

Remarks

The author of *object* can have *object* save the state when the **WriteProperties** event occurs, by calling the **WriteProperty** method of the **PropertyBag** object for each value that is to be saved.

Note: The *pb* property bag may be different from the *pb* that was passed to the most recent **ReadProperties** event.

The **WriteProperties** event may occur multiple times during the life of an instance of *object*.

WriteProperty Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthWritePropertyMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthWritePropertyMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthWritePropertyMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthWritePropertyMethodS"}

Writes a value to be saved to a **PropertyBag** class object.

Syntax

object.**WriteProperty(***DataName*, *Value*[, *DefaultValue*]**)**

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

Part	Description
object	An object expression that evaluates to an object in the **Applies To** list.
DataName	A string expression to represents the data value to be placed in the property bag.
Value	The data value to save in the property bag.
DefaultValue	The default value for the data.

Remarks

The **WriteProperty** method will write a data value in the property bag, and associate it with the string value in *DataName*. This string value will be used to access the data value when the **ReadProperty** method is called to retrieve a saved data value from the property bag.

Note: Specifying a default value reduces the size of the file belonging to the container of the control. A line for the property is written to the file only if the value to be written is different from the default. Wherever possible, you should specify default values for the properties of the control when initializing, saving, and retrieving property values.

Picture Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjPictureC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjPictureX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjPictureP;vbproPictureP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjPictureM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjPictureE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjPictureS"}

The **Picture** object enables you to manipulate bitmaps, icons, metafiles enhanced metafiles, GIF, and JPEG images assigned to objects having a **Picture** property.

Syntax

Picture

Remarks

You frequently identify a **Picture** object using the **Picture** property of an object that displays graphics (such as a **Form** object or a **PictureBox** control). If you have a **PictureBox** control named Picture1, you can set one **Picture** object equal to another using the **Set** statement, as in the following example:

```
Dim X As Picture
Set X = LoadPicture("PARTY.BMP")
Set Picture1.Picture = X
```

You can use an array of **Picture** objects to keep a series of graphics in memory without needing a form that contains multiple **PictureBox** or **Image** controls.

You can not create a **Picture** object using code like `Dim X As New Picture`. If you want to create a **Picture** object, you must use the **StdPicture** object like this:

```
Dim X As New StdPicture
```

Handle Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHandleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHandleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHandleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHandleS"}

Returns a handle to the graphic contained within a **Picture** object.

Syntax

object.**Handle**

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

Return Value

The value returned by the **Handle** property depends on the current setting of the **Type** property as shown in the following table:

Type Property	Return Value
1 (Bitmap)	An HBITMAP handle.
2 (Metafile)	An HMETAFILE handle.
3 (Icon)	An HICON or an HCURSOR handle.
4 (Enhanced Metafile)	An HENHMETAFILE handle.

Remarks

The **Handle** property is useful when you need to pass a handle to a graphic as part of a call to a function in a dynamic-link library (DLL) or the Windows API.

hPal Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbprohPalC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbprohPalX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbprohPalA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbprohPalS"}

Returns or sets a handle to the palette of a picture in a **Picture** object.

Syntax

object.**hPal** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	The handle to the palette for the picture (HPAL).

Remarks

The **hPal** property is useful when you need to pass a handle to a palette as part of a call to a function in a dynamic-link library (DLL) or the Windows API.

Render Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthRenderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthRenderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthRenderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthRenderS"}

Draws all or part of a source image to a destination object.

Syntax

object.**Render**(*hdc*, *xdest*, *ydest*, *destwid*, *desthgt*, *xsrc*, *ysrc*, *srcwid*, *srchgt*, *wbounds*)

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
hdc	Required. The handle to the destination object's device context.
xdest	Required. The x-coordinate of upper left corner of the drawing region in the destination object. This coordinate is in the scale units of the destination object.
ydest	Required. The y-coordinate of upper left corner of the drawing region in the destination object. This coordinate is in the scale units of the destination object.
destwid	Required. The width of drawing region in the destination object, expressed in the scale units of the destination object.
desthgt	Required. The height of drawing region in the destination object, expressed in the scale units of the destination object.
xsrc	Required. The x-coordinate of upper left corner of the drawing region in the source object. This coordinate is in HIMETRIC units.
ysrc	Required. The y-coordinate of upper left corner of the drawing region in the source object. This coordinate is in HIMETRIC units.
srcwid	Required. The width of drawing region in the source object, expressed in HIMETRIC units.
srchgt	Required. The height of drawing region in the source object, expressed in HIMETRIC units.
wbounds	Required. The world bounds of a metafile. This argument should be passed a value of Null unless drawing to a metafile, in which case the argument is passed a user-defined type corresponding to a RECTL structure.

Remarks

The recommended way to paint part of a graphic into a destination is through the **PaintPicture** method.

Setup Wizard

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

The Setup Wizard is a tool used with the Visual Basic Setup Toolkit that helps you create application setup and distribution media. You can also use the Setup Wizard to create dependency (.dep) files.

Note The Setup Wizard is designed for Visual Basic developers and isn't a general Windows Setup Tool.

Using the Microsoft Setup Wizard provides a number of options for the distribution of your application. The Setup Wizard supports:

- Multiple floppy disks, and can split files that are too large to fit onto a single floppy.
- Copying your files to a hard disk directory for distribution over a network or on CD-ROM.
- Distribution of your application across the Internet using automatic code download from Microsoft Internet Explorer, Version 3.0.

You can start the Setup Wizard by choosing either Application Setup Wizard from your Start menu or Setupwiz.exe from \Setupkit\Kitfil32 directory where you installed Visual Basic.

Based on information you provide, the Setup Wizard creates a special Setup.lst file. This information is then kept in the Vb5dep.ini file in your Windows directory.

Note The Setup Wizard is not a disk duplication tool. If you want to make copies of your master distribution disks, use a disk copying application.

Note When you create a new project Visual Basic adds some controls to the Toolbox and a reference to Data Access Objects (DAO). Setup Wizard recognizes the controls and references whether or not you use them in your application. If controls that you have not used in your application appear in the Confirm Dependencies step or if the Data Access step appears and you know you did not add any references to Data Access, you should clear the unused files. Either remove the tools and references manually from your source files or clear the checkbox for them in the file list.

Setup Wizard — Missing Dependencies

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbhowSetupWizardStep5C;vbproBooksOnlineJumpTopic"}

The Setup Wizard was unable to find dependency information for the files displayed in the list. Setup Wizard looks for dependency information in the Vb5def.ini and/or in special dependency files (.dep) that should exist for each EXE, OCX, and DLL in your application.

If an indicated file(s) comes from a third party, contact that party for dependency information or for the necessary dependency files (.dep).

If one of the files in the list is a project that you created in Visual Basic, you can use the Setup Wizard to create a dependency file for that project. Restart the Setup Wizard, select the project file (.vbp) for this component, and choose the Generate Dependency File Only option. Continue through all the steps to create the missing dependency file. Then, restart the Setup Wizard and try the current project again.

If you are certain that a file in the list does not need additional information, select it making sure that a check mark appears to the left of it. This will let the Setup Wizard know that this file has no dependencies, and the particular file will not appear in this dialog on your machine in the future.

Note If you leave some files unchecked, a dialog appears asking if you want the Setup Wizard to ignore the missing dependency information for this session. You can choose **Yes** and continue. You can also select not to see the dialog box in the future when you have missing dependency files.

Wizard Options

Missing Dependency Files List Lists the files for which the Setup Wizard can find no dependency files.

Help Displays the help topic for this step. You can also press F1 for help.

Cancel Cancels your previous actions and closes the Setup Wizard.

Back Moves you to the previous step.

Next Moves you to the next step.

Finish Creates a setup using the settings you selected and displays a confirmation dialog. Available only when you have provided enough information for the Setup Wizard to successfully create a setup package.

Setup Wizard — Finished!

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbhowSetupWizardStep8C;vbproBooksOnlineJumpTopic"}

The Setup Wizard has collected the information necessary to begin processing your distribution files.

The Setup Wizard notifies you when it has finished creating the distribution media.

Wizard Options

Save Template Displays the **Save Template As** dialog box where you can name and save your template in a location of your choosing. A template is a way of saving the current settings so that you can re-use them later. Load the template in the Select Project and Options step instead of a project (.vbp) file when you are asked to select a project.

Help Displays the help topic for this step. You can also press F1 for help.

Cancel Cancels your previous actions and closes the Setup Wizard.

Back Moves you to the previous step.

Next Disabled in this step.

Finish Creates a setup using the settings you selected and displays a confirmation dialog. Available only when you have provided enough information for the Setup Wizard to successfully create a setup package.

Remote Connection Details Dialog Box

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

Allows you to provide the settings necessary for the end user's machine to find and connect to a remote server. When end users run the setup program, they will be required to provide any information necessary for the connection that you did not enter here.

Dialog Box Options

Remote Support File Displays the name and path for the selected file. This is read-only, and is there for your convenience so you can see what file you are working on.

Remote Transport

Distributed COM (DCOM) — Your application uses DCOM to access the component.

Remote Automation — Your application uses Remote Automation to access the component.

Connection Information Provides information needed to connect to the component across the network.

Network Address — Lets you type the network address of the machine where the server application will be installed. If you leave this field blank, your users will be asked for this information when they execute setup.

Network Protocol — Displays a list of the network protocols from which you may choose. Available only if you selected Remote Automation. If you leave this field blank, your users will be asked for this information when they execute setup.

Authentication Available only if you chose Remote Automation. Required. For more information, see RPC Security in your Windows NT documentation.

Safety Dialog Box

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

Lists the components and lets you mark selected components as safe for initialization and/or safe for scripting. For Internet installations only. Making a component safe means that you claim it will not do harm to the end user's machine.

Warning By marking a component as safe for scripting or safe for initialization, you are guaranteeing that your component can never corrupt or cause undesirable behavior on an end user's machine, even if used in a Web page that you yourself did not author. Marking a component as either safe for scripting or safe for initialization may imply that you are accepting liability for this guarantee.

Note You must have your .cab files signed in order to take full advantage of these options. For more information about .cab files, see the *Programmer's Guide*.

Dialog Box Options

Components Lists the public UserControls, Classes, and/or UserDocuments within your ActiveX control, ActiveX EXE, and ActiveX DLL project. Select the item you want to mark as safe, and then select the appropriate option. You can select only one item at a time.

Note You must mark each item that you want to be safe. Repeat the process for each.

The selected component is:

Safe for initialization — Marks the component as safe for initialization. A control author should mark a control as "Safe for initializing" if, and only if, the control will not forsake security when initialized from *any* data, especially malicious data. This means the control will not produce any behavior that a reasonable user would deem unacceptable in the context of that user's expectations of the control's behavior. This is not a statement about the intentions of a control, but rather a promise from the control author that the control is impervious to damage attempted by malicious data. For example, a control marked "Safe for initializing" will not corrupt the end user's system regardless of data passed to the component during initialization. For controls marked "Safe for initializing", initialization data will not cause the control to create, change or delete arbitrary files, even temporary ones, or change system settings using this control.

Safe for scripting — Marks the component as safe for scripting. A control author should mark a control as "Safe for scripting" if, and only if, the control will not forsake security when scripted from *any* script, especially malicious scripts. This means the control will not produce any behavior that a reasonable user would deem unacceptable in the context of that user's expectations of the control's behavior. This is not a statement about the intentions of a control, but rather a promise from the control author that the control is impervious to damage attempted by malicious scripts. For example, a control marked "Safe for scripting", will not corrupt the end user's system no matter how it is scripted (how it is used programmatically in a Web page). For a control where users do not expect information to be collected about themselves or their personal computer, the control could not make such information available to a script. For controls marked "Safe for scripting", scripts should not be able to create, change or delete arbitrary files, even temporary ones, or change system settings using this control.

Note You can choose one or both of these options.

Help Displays help for the dialog box.

OK Accepts your selections and closes the dialog box.

Cancel Cancels your actions and closes the dialog box.

Setup Wizard — File Summary

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

Lists all the files you need to distribute with your application.

You should clear the checkbox of all files for which you do not have distribution rights. List in your release notes the files whose checkboxes you cleared so that your users can be sure they have all necessary files.

Wizard Options

File List Lists the files necessary for your application to function properly.

Add Adds a file(s) to the file list box. You can make multiple selections from the **Add Files** dialog box.

You can add a file to the distribution list, even if it is not required by the application. However, you cannot have duplicate filenames in this list even if they are located in different directories. If you try to add a duplicate file, a message box will notify you of the error.

Note If the Setup Wizard cannot find dependency information in Vb5dep.ini or an associated .dep file for the file you are trying to add, it displays a message box asking you if you still want to include the file. If you choose to include the file anyway, any dependency information needed by this file will not be included in the setup package and, as a result, your setup and application may not work properly. You can choose not to see this message for this file in the future.

If you know there are no dependencies, you can continue or you can return to the Select Project and Options screen and create a dependency file for this file.

File Details Use to display file size, file date and time, and current and destination location for the file you selected in the file list boxes. Version information is also displayed if available.

You can change the Installed Location for the selected file. However, changing the installed location of a shared file or remote ActiveX server may adversely affect its functionality.

If you are creating a setup for downloading from the Internet, the Destination Directory option is not available. Also, .vbl files can be downloaded from the Internet but are not registered. The end user must register the files using RegEdit.

Caution Changing the shared file status or the Destination Directory of a shared file or remote ActiveX server may adversely affect that component's capabilities and impair the removal of that component later. For example, if the same shared ActiveX component is installed to two different directories, the removal of one of them may result in removal of the registry information that makes each of them accessible, rendering the remaining component inaccessible.

Summary Details Displays a dialog box listing the number of program and setup files, the total number of files, the uncompressed bytes for each program and setup file, the total number of uncompressed bytes, and the path to the setup target.

Dependency Of Indicates the file that is needed by the selected file.

Help Displays the help topic for this step. You can also press F1 for help.

Cancel Cancels your previous actions and closes the Setup Wizard.

Back Moves you to the previous step.

Next Moves you to the next step.

Finish Creates a setup using the settings you selected and displays a confirmation dialog. Available only when you have provided enough information for the Setup Wizard to successfully create a setup package.

Setup Wizard — ActiveX Components

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

Provides you with the means for including additional ActiveX components in your project.

Note You may want to clear the checkbox for files that you know will already exist on the user's machine. You must also clear the checkbox of any files for which you do not have distribution rights. List in your release notes the files whose checkboxes you cleared in this screen so that users can be sure they have all necessary files.

Wizard Options

ActiveX Components List Displays a list of ActiveX components created when the Setup Wizard searched your project file (.vbp) and found references to ActiveX server components used in your application.

Add Local Displays the **Add Local** dialog box where you can locate and choose ActiveX components (*.dll, *.exe, *ocx) to the list of files to be distributed with your application. The **Add Local** dialog box is an **Open** common dialog box.

Add Remote Displays the **Add Remote** dialog box where you can locate and choose Remote ActiveX Components (*.vbr) to add to the list of files to be distributed with your application. The **Add Remote** dialog box is an **Open** common dialog box.

Remote Details Displays the **Remote Connection Details** dialog box where you can select Distributed COM (DCOM) or Remote Automation and add the network address, network protocol, and authentication information needed in order for the remote ActiveX component to function correctly.

File Details Displays the **File Details** dialog box where you can view information about your file.

Help Displays the help topic for this step. You can also press F1 for help.

Cancel Cancels your previous actions and closes the Setup Wizard.

Back Moves you to the previous step.

Next Moves you to the next step.

Finish Creates a setup using the settings you selected and displays a confirmation dialog. Available only when you have provided enough information for the Setup Wizard to successfully create a setup package.

Setup Wizard — Confirm Dependencies

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

Allows you to confirm additional file dependencies found by the Setup Wizard. The Setup Wizard lists any additional files, for example ActiveX controls used by or files referenced by your project.

Note You may want to clear the checkbox of any files that you know will already exist on the user's machine. You must also clear the checkbox of any files for which you do not have distribution rights. List in your release notes the files whose checkboxes you cleared so that users can be sure they have all necessary files.

Note When you create a new project, Visual Basic adds some controls to the Toolbox and a reference to Data Access Objects (DAO). Setup Wizard recognizes the controls and references whether or not you use them in your application. If controls that you have not used in your application appear in the Confirm Dependencies step or if the Data Access step appears and you know you did not add any references to Data Access, you should clear the unused files. Either remove the tools and references manually from your source files or clear the checkbox for them in the file list.

Wizard Options

File with dependencies list Lists the dependencies found for your application.

File Details Displays the **File Details** dialog box.

Help Displays the help topic for this step. You can also press F1 for help.

Cancel Cancels your previous actions and closes the Setup Wizard.

Back Moves you to the previous step.

Next Moves you to the next step.

Finish Creates a setup using the settings you selected and displays a confirmation dialog. Available only when you have provided enough information for the Setup Wizard to successfully create a setup package.

Setup Wizard — Data Access

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

The Setup Wizard has determined that you are using Data Access Objects (DAO) and lets you select any ISAM database format(s) you are using in your application. The appropriate ISAM engine files will be included in your setup program. You can also specify the type of workspace, Jet or ODBCDirect, that your application uses.

Note When you create a new project, Visual Basic adds some controls to the Toolbox and a reference to Data Access Objects (DAO). Setup Wizard recognizes the controls and references whether or not you use them in your application. If controls that you have not used in your application appear in the Confirm Dependencies step or if the Data Access step appears and you know you did not add any references to Data Access, you should clear the unused files. Either remove the tools and references manually from your source files or clear the checkbox for them in the file list.

Wizard Options

Installable ISAM's Lists available ISAM database formats you want to include. When you select one or more of these formats, a special file(s) is added to your distribution list that supports this format.

What type of Workspaces are you using? Determines which components you ship with Data Access Objects (DAO).

dbUseJet — Includes the Jet database files with your setup. If you choose any of the ISAM formats listed in the Installable ISAMs list, dbUseJet is automatically selected and cannot be cleared because, by using an ISAM (restrictive clause), you need to include the Jet engine files.

dbUseODBC — Includes the Remote Data Objects (RDO) files with your setup. If you use the **ODBCDirect** features of Data Access Objects (DAO), select this option.

Note You cannot clear both of these options. You must have one or both of them selected to proceed to the next step. If your application uses both the Jet engine and **ODBCDirect**, choose both options. Jet and RDO will be included in your setup. If you do not access any data through the Jet engine, and you use **ODBCDirect** exclusively, choose only dbUseODBC.

Help Displays the help topic for this step. You can also press F1 for help.

Cancel Cancels your previous actions and closes the Setup Wizard.

Back Moves you to the previous step.

Next Moves you to the next step.

Finish Creates a setup using the settings you selected and displays a confirmation dialog. Available only when you have provided enough information for the Setup Wizard to successfully create a setup package.

Setup Wizard — Distribution Method

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

Allows you to choose the way you want to distribute your application.

Wizard Options

Floppy Disk Copies setup.exe and all of the setup files for your project onto floppy disks.

Single Directory Copies setup.exe and all of the setup files into a single directory from which your users can run setup.

Disk Directories Creates multiple directories such as \Disk1, \Disk2, and \Disk3, and copies the files into appropriate directories as if each directory represented a floppy disk in your setup. Your users can run setup from \Disk1 or make floppies from the disk images.

- You should delete all subdirectories and files in the targeted directory using the Windows Explorer or File Manager before continuing. Setup Wizard does not do this for you.
- You cannot choose the root drive of a network or local drive unless it is a floppy drive.

Help Displays the help topic for this step. You can also press F1 for help.

Cancel Cancels your previous actions and closes the Setup Wizard.

Back Moves you to the previous step.

Next Moves you to the next step.

Finish Creates a setup using the settings you selected and displays a confirmation dialog. Available only when you have provided enough information for the Setup Wizard to successfully create a setup package.

Setup Wizard — Floppy Disk

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

Gathers information about the local destination and disk size for an application that you want to distribute on floppy disks.

Wizard Options

Floppy Drive Lists only the disk drives that appear on the current machine.

Disk Size Lists the available disk sizes.

Help Displays the help topic for this step. You can also press F1 for help.

Cancel Cancels your previous actions and closes the Setup Wizard.

Back Moves you to the previous step.

Next Moves you to the next step.

Finish Creates a setup using the settings you selected and displays a confirmation dialog. Available only when you have provided enough information for the Setup Wizard to successfully create a setup package.

Setup Wizard — Internet Distribution Location

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

Gathers information about the directory where you want the Setup Wizard to place the Cabinet (.cab) file it creates, HTML sample code, any documentation, and additional files, and where it should create the \Support sub-directory.

Note You have to copy the .cab files from this directory to your web site.

Note The generated .cab file will *not* contain all of the runtime support files required to run your component. The .cab file contains an .inf file which points to the required runtime components and downloads them only as necessary. For example, it will download vbrun500.cab. The runtime .cab files are available on the Microsoft Web site and may be automatically downloaded from there, or can be moved to a location within your intranet. The benefit of this approach is that the Setup Wizard-generated .cab file downloads faster if it does not contain all of the support files that may already exist on the end user's machine.

The Setup Wizard creates .cab files from ActiveX Control projects and ActiveX EXE and ActiveX DLL projects that have public classes. This includes projects that contain **UserDocuments**. The .cab files can be automatically downloaded, expanded, and installed by Microsoft Internet Explorer, version 3.0, when referenced from an HTML page using the CODEBASE tag.

Setup Wizard does the following:

- Creates a .cab file for the project's components, leaving additional space to let you digitally sign your .cab files. This file will be in the directory you specify. For more information on digitally signing see the *ActiveX SDK*.
- Creates an .inf file, used within the .cab file, which describes how to download and register the project's components and describes where and how to download other .cab files containing the VB runtime, VB supplied ActiveX controls, and other runtime components such as DAO.
- Creates a sample HTML template which includes the OBJECT and CODEBASE tags for the given component.
- Creates a subdirectory off of the user-specified directory (\Support) which contains all of the components (namely the .inf and .ddf files) built into the .cab so users can make modifications and regenerate the .cab file on their own if necessary.
- Provides means for specifying that components are Safe for Scripting and Safe for Initialization. Places appropriate information into the .inf file so that registry information will be written during the installation process.

Setup Wizard does not automatically create the .lpk files that are needed if your component requires licensing. You can use LPK_tool.exe in the \Tools directory on your Visual Basic CD to do this.

Note If you try to place files in a directory that is not empty, Setup Wizard tells you that there are files and/or subdirectories in that directory. You will be asked if you want to continue. If you choose **Yes** to continue, Setup Wizard will place the files in that location. Even if you are creating multiple disks, the files with the same name will be overwritten, but other existing files will not be automatically deleted. It is recommended that you delete the contents of the directory before proceeding.

Wizard Options

Destination Lists the destination of the directory where you want the installation files to be placed. First you select the drive where you want your files placed. Then you select the specific directory. If the directory does not exist, Setup Wizard will ask you if you want to create one. The last box allows you to see the tree structure of your path.

Note Use the directory list box to change the path by double-clicking on the drive or folder where you want the files to go.

Help Displays the help topic for this step. You can also press F1 for help.

Cancel Cancels your previous actions and closes the Setup Wizard.

Back Moves you to the previous step.

Next Moves you to the next step.

Finish Creates a setup using the settings you selected and displays a confirmation dialog. Available only when you have provided enough information for the Setup Wizard to successfully create a setup package.

Setup Wizard — Introduction

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

Helps you create a setup a program and/or dependency files for your application.

Wizard Options

Skip this screen in the future When selected, the Setup Wizard will not show the Introduction screen the next time you start it.

If you want to see this screen in the future, click the **Back** button on the first step of the wizard.

Help Displays the help topic for this step. You can also press F1 for help.

Cancel Cancels your previous actions and closes the Setup Wizard.

Back Disabled in this step.

Next Moves you to the next step.

Finish Creates a setup using the settings you selected and displays a confirmation dialog. Available only when you have provided enough information for the Setup Wizard to successfully create a setup package.

Setup Wizard — Out of Date Dependencies

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

Lists the file whose dependency location contains information that thinks the file is out of date. For example, it lists a file whose information in the dependency file for a file that is older than the current file. The Setup Wizard uses version information to determine this. If version information cannot be found, the Setup Wizard uses the date of the file in the format mm/dd/yyyy H/mm/ss to evaluate for out of date dependencies.

Note The version information in the dependent file must be an exact match. Either older or newer versions are considered out of date. When the Setup Wizard uses the date information, the newer date is used for the match.

Having out of date dependency information can cause setup and your application not to work correctly. If you choose not to rebuild files with out of date dependencies, a dialog box appears asking you to confirm that you want the Setup Wizard to continue even though the dependency information is out of date. You can also choose not to see the dialog box in the future.

Note If you see this dialog, one or more dependency files (.dep) are out of date with respect to a component in the list. This means that the component is newer than the version recorded in the dependency file. Since it is possible for the component's set of dependency information to have changed since the dependency file was created, the Setup Wizard cannot be certain that this dependency information is correct.

If the indicated file(s) comes from a third party, contact that party for updated dependency files (.dep).

If the indicated file(s) is a Visual Basic component that you created, restart the Setup Wizard, select the project file for this component and the Generate Dependency File Only option, and continue through all the steps to create the out-of-date dependency file. Then, restart the Setup Wizard and try the current project again.

Wizard Options

Files With Out of Date Dependencies List Lists the files that have dependency information that is out of date.

Help Displays the help topic for this step. You can also press F1 for help.

Cancel Cancels your previous actions and closes the Setup Wizard.

Back Moves you to the previous step.

Next Moves you to the next step.

Finish Creates a setup using the settings you selected and displays a confirmation dialog. Available only when you have provided enough information for the Setup Wizard to successfully create a setup package.

Setup Wizard — Select Project and Options

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

Allows you to choose the Visual Basic project (.vbp) file of, or Setup Wizard template (.swt) file for, the application that you want to distribute and lets you choose to create a setup program – with or without a dependency file –, an Internet download setup, or a dependency file only. The Setup Wizard reads the .vbp and source files for information on which objects, references, resources, dynamic-link libraries, and so forth need to be included in the distribution files, as well as the location of your .exe or .dll file.

Tip It is good development practice to always provide a dependency (.dep) file with your application if it is a shared component such as an ActiveX control or ActiveX component, even if the dependency file simply indicates that there are no dependencies for the component other than the Microsoft Visual Basic runtime files.

Wizard Options

Project Location Lets you type the name of the project file you want to distribute and its location on your machine. If the executable (.exe, .dll, or .ocx) file doesn't exist for the specified application, the Setup Wizard automatically builds it.

Browse Displays the standard **Open** common dialog box where you can locate and choose the project file of the application that you want to build. The selected filename is automatically inserted in the Project Location field.

Rebuild the Project When selected, forces a rebuild of the project's .exe, .dll. or .ocx file whether or not Setup Wizard finds an existing .exe, .dll, .ocx file for your project.

Choose this option if you want to rebuild before distributing your application. It is selected by default and it is recommended that you leave it selected.

Note You must build your project and save the project file at least once before using the Setup Wizard. Not doing so may cause Setup Wizard to fail and your setup distribution may not work properly.

Options

- Create a Setup Program

  Generate Dependency File — Generates a file that contains dependency information and includes it in the setup.

  Dependency files are used for any object of an .ocx, .dll, or .exe, ActiveX component, or a project that could be used as a component in other projects.

  The default install location for the .dep files is the same directory as the install location of the .ocx. You can change the install location for the .dep in the **File Details** dialog box from the File Summary step.

  Note Changing the location of the .dep file may prevent Setup Wizard from finding it when it is part of a subsequent Setup Wizard session on a project using this component.

  Note You cannot have two files with the same base name that require dependency files. Each file must have a unique dependency file and by default the dependency file uses the same name as the file, but adds to it the .dep extension.

- Create Internet Download Setup — Allows you to create an Internet download setup for only ActiveX Control projects and ActiveX EXE, and ActiveX DLL projects that have public classes, including projects that contain UserDocuments. Not available in the Visual Basic Learning Edition.

  Note When you select this option, a **What's New** button appears. Clicking **What's New** displays a Microsoft Web page where you can view and download current information on Internet download.

- Generate Dependency File Only — Generates a dependency file with the same name as your

project and a .dep extension, and places it in the same directory as your project.

To correctly create the dependency file, you must provide accurate information (as if you were actually creating a setup for the application) for all of the remaining steps of the Setup Wizard. The dependency file will contain dependency information, for example, dependent files, their destinations, and their versions, necessary to install the project as part of another application.

What's New Takes you to the Microsoft Web site where you can browse for additional information about the Internet download.

Only available if you chose Create Internet Download Setup.

Help Displays the help topic for this step. You can also press F1 for help.

Cancel Cancels your previous actions and closes the Setup Wizard.

Back Moves you to the previous step.

Next Moves you to the next step.

Finish Creates a setup using the settings you selected and displays a confirmation dialog. Available only when you have provided enough information for the Setup Wizard to successfully create a setup package.

Setup Wizard — Shared ActiveX Application

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

Determines whether to distribute your application as a separate application in its own directory on the end user's machine, or as a shared ActiveX component. If installed as a shared ActiveX component, this step allows you to determine whether this component will be installed onto a remote server machine accessed through Remote Automation.

Wizard Options

Install as a stand-alone application Installs the file as a stand-alone file into the application directory that the end user will specify.

Install as a shared component Installs the component as a shared ActiveX component into the \Windows\System directory. This is the default.

Will this shared component be installed on a remote server and accessed through Remote Automation?

Yes — Application is installed on a remote server and accessed through Remote Automation.

No — Application is not accessed through Remote Automation. If your application is accessed remotely through Distributed COM (DCOM), you should select No.

Help Displays the help topic for this step. You can also press F1 for help.

Cancel Cancels your previous actions and closes the Setup Wizard.

Back Moves you to the previous step.

Next Moves you to the next step.

Finish Creates a setup using the settings you selected and displays a confirmation dialog. Available only when you have provided enough information for the Setup Wizard to successfully create a setup package.

Setup Wizard — Single Directory

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

Gathers information the Setup Wizard needs to create a setup program that can be installed from a single directory.

Note If you try to place files in a directory that is not empty, Setup Wizard tells you that there are files and/or subdirectories in that directory. You will be asked if you want to continue. If you choose **Yes** to continue, Setup Wizard will place the files in that location but will not delete any files. Files with the same name will be overwritten. You will have to delete other files yourself if you want to remove them.

Wizard Options

Destination Lists the destination of the directory where you want to place the setup program and installation files. First you select the drive where you want your files placed. Then you type in the specific directory. If the directory does not exist, Setup Wizard will ask you if you want to create one. The last box allows you to see the structure of your path.

Note Use the directory list box to change the path by double-clicking on the drive or folder you want to use as your destination.

Help Displays the help topic for this step. You can also press F1 for help.

Cancel Cancels your previous actions and closes the Setup Wizard.

Back Moves you to the previous step.

Next Moves you to the next step.

Finish Creates a setup using the settings you selected and displays a confirmation dialog. Available only when you have provided enough information for the Setup Wizard to successfully create a setup package.

Setup Wizard — Multiple Directories

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

Gathers information the Setup Wizard needs to create a setup program that can be installed from multiple directories, or that can be copied from these directories to multiple floppy disks.

Note If you try to place files in a directory that is not empty, Setup Wizard tells you that there are files and/or subdirectories in that directory. You will be asked if you want to continue. If you choose **Yes** to continue, Setup Wizard will place the files in that location but will not delete any files. Files with the same name will be overwritten. You will have to delete other files yourself if you want to remove them.

Wizard Options

Destination Lists the destination of the directory where you want to install the application. First you select the drive where you want your files placed. Then you type in the specific directory. If the directory does not exist, Setup Wizard will ask you if you want to create one. The last box allows you to see the tree structure of your path.

Note You can use the directory list box to change the path by double-clicking on the drive or folder you want to use as your destination.

Disk Size Lists the available disk sizes.

Help Displays the help topic for this step. You can also press F1 for help.

Cancel Cancels your previous actions and closes the Setup Wizard.

Back Moves you to the previous step.

Next Moves you to the next step.

Finish Creates a setup using the settings you selected and displays a confirmation dialog. Available only when you have provided enough information for the Setup Wizard to successfully create a setup package.

INF File

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

Lists the components required for installing the ActiveX components on your user's machine. The .inf file contains:

- A list of all file dependencies needed for Internet download setup.
- A list of linked .cab files and their locations.
- Version information.
- Class IDs for components.
- Platform specific information.
- Destination directory specifics.
- Registry information for writing Safe for Initialization and Scripting entries.

Confirm Missing Dependencies Dialog Box

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

Confirms that you really do want to ignore the missing dependencies.

Dialog Box Options

Yes Tells the Setup Wizard that you do want to proceed without the missing dependencies. Any additional files that may be needed may not be installed. If you choose **Yes**, any additional files that may be needed by the unselected files will not be installed. This may result in setup and your application not functioning properly.

If you are certain that a particular file does not have any dependencies and does not need to be registered in the system registry to function correctly, you should choose **No**.

If the indicated file(s) comes from a third party, contact that party for dependency information or for the necessary dependency files (.dep).

If the indicated file(s) is a Visual Basic component that you created, restart the Setup Wizard, select the project file for this component and the Generate Dependency File Only option, and continue through all the steps to create the missing dependency file. Then, restart the Setup Wizard and try the current project again.

No Tells the Setup Wizard that you want to correct the missing dependencies.

Do not show this dialog again When selected, this dialog never appears again.

Confirm Out of Date Dependencies Dialog Box

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

Confirms that you really do want to ignore the dependency information that is out of date.

Dialog Box Options

Yes Tells the Setup Wizard that you do want to proceed with the out of date dependency information.

If you choose **Yes**, Setup Wizard attempts to use the out of date dependency information to build your setup program. This may mean that needed files will not be installed. As a result, setup and your application may not function properly.

If you are certain that the out of date dependency file contains the correct dependency information, choose **Yes** and continue.

If the indicated file(s) comes from a third party, contact that party for dependency information or for the necessary dependency files (.dep).

If the indicated file(s) is a Visual Basic component that you created, restart the Setup Wizard, select the project file for this component and the Generate Dependency File Only option, and continue through all the steps to create the missing dependency file. Then, restart the Setup Wizard and try the current project again.

No Tells the Setup Wizard that you do not want to proceed but want to update the dependency information. This is the recommended choice.

Do not show this dialog again When selected, this dialog never appears again.

File Details for ActiveX Server Component Dialog Box

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

Displays the following information about your ActiveX server component:

- File information such as the file size and date it was created.
- Installation information is not available at this time because it has not been determined yet.
- Version information such as your company name, a description of the file, the version number, legal copyrights and trademarks, the product name and the product version number. The version information is displayed only when Setup Wizard can find it in the selected file.

OK Closes the dialog box.

Help Displays the help topic for this dialog. You can also press F1 for help.

File Details for Confirm Dependencies Dialog Box

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

Displays the following information for the dependencies:

- File information such as the file size and date it was created.
- Installation information is not available at this time because it has not been determined yet.
- Version information such as your company name, a description of the file, the version number, legal copyrights and trademarks, the product name and the product version number. The version information is displayed only when Setup Wizard can find it in the selected file.

OK Closes the dialog box.

Help Displays the help topic for this dialog. You can also press F1 for help.

File Details for File Summary Dialog Box

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

Displays the following information for the selected file and allows you to change the installation information.

File information File size and date the file was created.

Installation information

Not available if you are creating an Internet Download setup.

Destination Directory — You can type or choose the directory where this file will be installed on the end user's machine.

Note If you are creating a setup for downloading from the Internet, the Destination Directory option is not available.

The format for the destination directory starts with a predefined macro and has optional subdirectories. For example, if the Program Files directory is located at C:\Program Files, you could type

$(ProgramFiles)*MyCompany\MyApplication*

to install the program files into the C:\Program Files\MyCompany\MyApplication directory. You can choose to enter an absolute path and ignore the pre-defined macros, but it is not recommended that you do so.

Install as a shared file — Indicates whether or not to install the file as a system shared component. If selected, the file will be removed only when the last program to use this file is uninstalled.

Register contained license information on the end user's machine. — Available only if the file is a Visual Basic ActiveX control license file (.vbl). If selected, the license information contained in the file will be registered on the end user's machine. A .vbl file is created by Visual Basic for ActiveX control projects if you selected the Require License Key option on the General tab of the **Project Properties** dialog box.

Note If you have a Visual Basic license file (.vbl), Setup Wizard adds an option, **Do not install this file**, to the Destination Directory list and selects it by default. The license file is registered but not installed on the end user's machine. You can override the default and force the file to be installed by changing the Destination Directory. Generally, you want the user to have the license file registered, but not installed on the machine, so that the user can use controls in an application and distribute the application with only the runtime version of the controls. The end user can register the .vbl files using RegEdit.

Version information Your company name, a description of the file, the version number, legal copyrights and trademarks, the product name and the product version number. The Version information is displayed only when Setup Wizard can find it.

Caution Changing either the shared file status or the Destination Directory of a shared file or remote ActiveX server may adversely affect the component's capabilities and impair the removal of that component later. For example, if the same shared ActiveX component is installed to different directories, the removal of one of them may result in removal of the registry information that makes each of them accessible, rendering the remaining component inaccessible.

OK Accepts the changes and closes the dialog box.

Cancel Closes the dialog box without accepting any changes.

Help Displays help for this dialog box.

The ActiveX Component you selected is not Self-Registerable.

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

You were attempting to add an ActiveX component using the **Add** button. However, the component that you selected does not appear to be capable of registering itself in the system registry.

For an ActiveX EXE server to be added in this step, it needs to support the command-line parameters, /RegServer and /UnRegServer, which tell the component to self-register and unself-register itself and then to do so silently.

For an Active X DLL server to be added in this step, it needs to support the DllRegisterServer and DllUnregisterServer DLL entry points.

The component that you selected does not contain the **OLESelfRegister** keyword in its version information. This keyword is normally built into components that are capable of self-registration. Some components, however, are capable of self-registration but do not have this keyword in their version information.

Yes Tells Setup Wizard to add the component to the setup and cause it to be self-registered when installed. Choose **Yes** if you are certain that the component you selected supports the command-line parameters or the DLL entry points described here. If you choose **Yes**, but the component does not support self-registration, you may have errors in your setup application.

No Tells Setup Wizard not to add the component at this step. You can add the file in the File Summary step, in which case it will be installed but not self-registered. You may need to modify your setup program by modifying the setup1.vbp project manually in your \Vb\Setupkit\Setup1 subdirectory to cause the necessary registration information to be added to the end user's machine.

Help Displays help for this box.

Setup Wizard — Internet Package

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

Allows you to set options regarding the creation of the Internet package. When you choose to create an Internet Download setup, Setup Wizard creates a single .cab file, known as the primary .cab file, that contains the components for VB5-created ActiveX controls , ActiveX DLL, ActiveX EXEs, and ActiveX documents and links them to at least one other .cab file for the Visual Basic runtime.

A .cab file includes:

- The project components such as the ActiveX control, ActiveX DLL, or ActiveX EXE.
- The .inf file which contains safe for scripting and safe for initialization registry information as defined by the user, and links to other .cab files that contain VB support files and controls. This file replaces the setup.lst file that the Setup Wizard creates in the standard setup.
- Reserved space for digital signatures.
- All files that are not in other .cab files.

For ActiveX Control projects, ActiveX EXEs, and ActiveX DLLs, all runtime components such as vbrun500.dll, individual controls, Data Access Objects (DAO), and Remote Data Objects (RDO) are packaged into separate .cab files, digitally signed by Microsoft, and placed on the Microsoft web site. You can choose to link your files to the .cab files on the Microsoft web site or you can download local copies of them.

The benefit of using .cab files from an Internet web site are:

- You do not need to distribute all of the .cab files required by your application. The only file you need to distribute is the primary .cab file.

  The .inf file within the primary .cab file points to the Microsoft web site and downloads the necessary .cab files based on the needs of the end user.

- They provide an efficient means of delivering updates to your product.

If you cannot or do not want to connect to the Internet, you may place the .cab files on a server within your intranet. This allows for faster downloading while allowing users to remain on a secured web.

Although Visual Basic, version 5.0, does not include any tools to digitally sign components, the Setup Wizard-generated .cab files reserve space for you to add a digital signature at a later time. To get your files signed you can:

- Send them to a service provider such as VerSign, who keeps the private key. The service provider signs the files and returns them to you.
- Purchase special hardware and/or software which stores your private key locally. When you are ready to sign your files, run a utility such as CodeSign to sign your files. The *ActiveX SDK* includes tools to create text signatures.
- Use the *ActiveX SDK* to create test digital signatures.

Visual Basic, version 5.0, does not automatically create .lpk files which are required for components that require licensing if you plan to use them on HTML pages. The Setup Wizard works on a Visual Basic project level to create the necessary download components. The .lpk file must contain all of the licensing information for all of the components on a given HTML page. You can run LPK_Tool.exe from the \Tools directory on your Visual Basic CD after inserting all of the required ActiveX components on a given HTML page.

Wizard Options

Runtime Components

Download from the Microsoft web site — Links the files to the .cab files found on the Microsoft web site or any other web sites defined within the dependency (.dep) file. For example, a company

could have a new .dep file for their component which points to their web site. The Setup Wizard writes the appropriate information into the .inf file in the .cab file it creates.

Use Local Copy — Links the files to a local copy of the .cab files. Requires you to enter the URL or UNC where you have downloaded the .cab files or to leave the local copy box blank if the .cab files will be in the same directory as the primary .cab file that the Setup Wizard creates. Blank is the Default.

This information is placed in the .inf file. If you put invalid information in the local copy box, you can edit the .inf file that is located in the \Support directory and manually rebuild the .cab file using Diantz.exe. See your *AciveX SDK* for additional information.

Safety Displays the **Safety** dialog box where you can determine which files are safe for initialization and/or safe for scripting. Microsoft Internet Explorer, version 3.0, has default security that requires that before any ActiveX component is instantiated on an HTML page, it must be marked Safe for Scripting and Safe for Initialization.

Help Displays the help topic for this step. You can also press F1 for help.

Cancel Cancels your previous actions and closes the Setup Wizard.

Back Moves you to the previous step.

Next Moves you to the next step.

Finish Creates a setup using the settings you selected and displays a confirmation dialog. Available only when you have provided enough information for the Setup Wizard to successfully create a setup package.

Summary Details Dialog Box

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

Displays the summary details for the installation. It lists the following information:

- Number of program files Total number of user files that will be installed.
- Number of setup files Total number of additional files required by the installation, for example setup.exe.
- Total number of files Total number of files required by the installation.
- Uncompressed bytes of program files Total number of types of user files to be installed by the installation before compression.
- Uncompressed bytes of setup files Total number of types of additional files required by the installation before compression.
- Total uncompressed bytes of all files Total number of types to be installed before compression.
- Setup Target The path into which the distribution files will be placed when the Setup Wizard is finished.

Unable To Locate Dependency Information Dialog Box

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

Displays the name of a file for which no dependency information could be found.

The Setup Wizard has determined that the file is missing dependency information. This means that you are missing one or more dependency files (.dep) which provide information about other files that are needed in order to correctly install a component.

If an indicated file(s) comes from a third party, contact that party for dependency information or for the necessary dependency files (.dep).

If the indicated file(s) is a Visual Basic component that you created, restart the Setup Wizard, select the project file for this component and the Generate Dependency File Only option, and continue through all the steps to create the missing dependency file. Then, restart the Setup Wizard and try the current project again.

If you are certain that a file in the list does not need additional files to run, select it making sure that a check mark appears to the left of it. This will let the Setup Wizard know that this file has no dependencies, and the particular file will not appear in this dialog in the future.

Note If you leave some files unchecked, a dialog appears asking if you want the Setup Wizard to ignore the missing dependency information. You can choose **Yes** and continue. You can also select not to see the dialog box in the future when you have missing dependency files.

Dialog Box Options

Yes Include the file anyway.

No Do not include the file.

Help Displays a help topic for this dialog box.

Never warn me about this file's dependency information again If selected does not show this dialog box again. You should choose this option only if you are certain that this file requires no other files and does not need to be registered in the system registry to function properly.

Macros for Installing Files

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

You can use macros in the Setup Wizard to indicate a path specified by a user. Listed below are macros that can be used in the **Destination Directory** box of the **Files Details for File Summary** dialog box.

Note The macro you choose is usually used in the DefaultDir value in Setup.lst, and should not be used elsewhere. $(AppPath) or subdirectories of $(AppPath) should be the typical installation destination for application files, since this allows the user to change the installation location.

Macro	Installs into the
$(WinSysPath)	\Windows\System subdirectory under Windows 95 or the \Windows\System32 directory under Windows NT.
$(WinPath)	\Windows directory.
$(AppPath)	Application directory specified by the user, or the DefaultDir value specified in the [SETUP] section of Setup.lst.
$(AppPath)samples	\Samples subdirectory below the application directory.
C:*path*	Directory identified by *path*. Not recommended.
$(CommonFiles)	Common directory to which shared files are sometimes installed: c:\Program Files\Common Files\ (Windows 95, Windows NT, Version 4.0). c:\Windows (Window NT, Version 3.51).
$(ProgramFiles)	Directory to which applications are usually installed: c:\Program Files (Windows 95 Windows NT, Version 4.0). c:\(Windows NT, Version 3.51).
$(CommonFiles)\System	Subdirectory, System, of the common directory to which shared files are sometimes installed. Same as $(CommonFiles) with a subdirectory indicated.
$(MSDAOPath)	Location that is stored in the registry for Data Access Objects (DAO) components. You should not use this for your files.

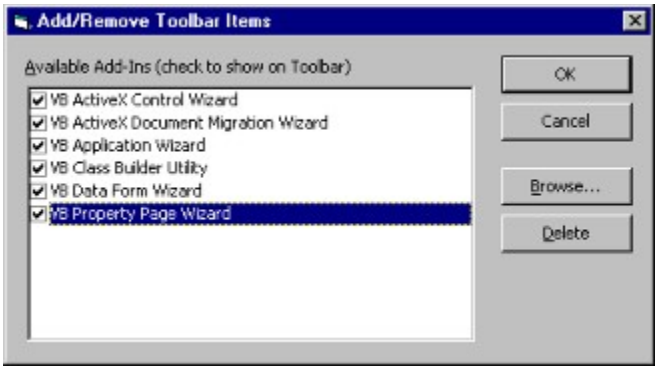
Add-In Toolbar

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

A toolbar on which to place add-ins and Wizards for quick and easy user access. To start an add-in or Wizard, simply click its icon on the toolbar.

The add-ins and Wizards placed on the Add-In toolbar are not activated until their button is clicked. The Add-In toolbar eliminates the need for activating the add-in through the **Add-In Manager** dialog box.

You can add Wizards and add-ins to the Add-In toolbar through the **Add/Remove Toolbar Items** (+/-) button. When you click this button, you get the following dialog box:



To add an add-in or Wizard to the list of available add-ins, click the **Browse** button. Point to an add-in or Wizard's .Exe or .Dll file in the dialog box, then click **Open**. It should appear in the **Available Add-Ins** list. It will not show up on the Add-In toolbar, however, unless its box is checked in the **Available Add-Ins** list.

The **OK** button closes the **Add/Remove Toolbar Items** dialog box and updates the Add-In toolbar with the checked items.

The **Cancel** button closes the **Add/Remove Toolbar Items** dialog box and ignores any changes made when it was opened.

When you click the **Delete** button, the currently selected add-in or Wizard is removed from the **Available Add-Ins** list. Note that this does not remove the add-in or Wizard from the system, nor its reference in the **Add-In Manager** dialog box. The **Delete** button removes only the entry in the Add-In toolbar **Available Add-Ins** list.

AddIns Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAddInsPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAddInsPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAddInsPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAddInsPropertyS"}

Returns a collection which add-ins can use to register their automation components into the extensibility object model.

Syntax

object.**AddIns**

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

AddToAddInToolbar Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthAddToAddInToolbarMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthAddToAddInToolbarMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthAddToAddInToolbarMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthAddToAddInToolbarMethodS"}

Inserts a button on the Add-In toolbar which references an add-in or Wizard.

Syntax

object.**AddToAddInToolbar** (*sfilename* **As String**, *sprogid* **As String**, *showontoolbar* **As Boolean**, *forceaddintoolbar* **As Boolean**)

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
sfilename	Required. A string expression specifying the path to the add-in or Wizard and the name of its .Exe or .Dll file.
sprogid	Required. A string expression specifying the programmatic ID (ProgID) of the add-in or Wizard.
saddinname	Required. A string expression specifying the title of the add-in or Wizard.
showontoolbar	Required. A Boolean expression specifying whether the add-in or Wizard referred to will appear on the Add-In toolbar. **True** = yes, **False** = no.
forceaddintoolbar	Required. A Boolean expression specifying whether the Add-In toolbar is automatically displayed the next time Visual Basic is started. **True** = yes, **False** = no.

AddToAddInToolbar Method Example

This example uses the **AddToAddInToolbar** method to add a button to the Add-In toolbar for a fictitious add-in called MyAdd.Dll. Setting ForceAddInToolbar to **True** ensures that the Add-In toolbar is loaded the next time Visual Basic is started.

You could modify the following in a small Visual Basic application to serve as a Setup for your add-in.

```
Sub Main()
 dim x as Object
 Set x=CreateObject("AddInToolbar.Manager")
 x.AddToAddInToolbar sFileName:="C:\VB5\MyAdd.DLL", _
    sProgID:="MyAddIn.Connect", _
    sAddInName:="MyAddIn Title" _
    ShowOnToolBar:=True, _
    ForceAddInToolbar:=True
End Sub
```

AsyncProperty Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjAsyncPropertyObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjAsyncPropertyObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjAsyncPropertyObjectP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjAsyncPropertyObjectM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjAsyncPropertyObjectE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjAsyncPropertyObjectS"}

The **AsyncProperty** object is passed in to the AsyncReadComplete event and contains the results of the **AsyncRead** method.

AsyncType Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAsyncTypePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAsyncTypePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAsyncTypePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAsyncTypePropertyS"}

Returns or sets the type of the data returned by the **Value** property. This property is available only as an argument of the **AsyncRead** method.

Syntax

object.**AsyncType** = *dataType*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
dataType	An integer specifying the data type, as shown in Settings below.

Settings

The settings for *dataType* are:

Constant	Value	Description
vbAsyncTypePicture	0	Default. Picture object.
VbAsyncTypeFile	1	The data is provided in a file created by Visual Basic.
VbAsyncTypeByteArray	2	The data is provided as a byte array that contains the retrieved data.

Bindable Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBindablePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBindablePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBindablePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBindablePropertyS"}

Returns or sets the **Bindable** property associated with a **Member** object.

Syntax

object.**Bindable**

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

Browsable Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBrowsablePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBrowsablePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBrowsablePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBrowsablePropertyS"}

Returns or sets the Browsable attribute associated with a **Member** object.

Syntax

object.**Browsable**

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

Category Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproCategoryPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCategoryPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCategoryPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCategoryPropertyS"}

Returns or sets the Category attribute associated with a **Member** object.

Syntax

object.**Category**

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

Connect Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproConnectPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproConnectPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproConnectPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproConnectPropertyS"}

Returns or sets the connected state of an add-in.

Syntax

object.**Connect**

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

ContainedVBControls Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcolContainedVBControlsCollectionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbcolContainedVBControlsCollectionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbcolContainedVBControlsCollectionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcolContainedVBControlsCollectionS"}

The ContainedVBControls collection represents a collection of **VBControl** objects.

ContinuousScroll Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproContinuousScrollPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproContinuousScrollPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproContinuousScrollPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproContinuousScrollPropertyS"}

Returns or sets a value that determines if scrolling is continuous, or if the **UserDocument** only redraws when the scroll thumb is released.

Syntax

object.**ContinuousScroll** = *boolean*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that specifies whether scrolling is continuous or not.

Settings

The settings for boolean are:

Setting	Description
True	Default. Scrolling is continuous.
False	The UserDocument redraws only when the thumb is released.

Controls Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproControlsPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproControlsPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproControlsPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproControlsPropertyS"}

Returns a reference to a collection of **Control** objects.

Syntax

object.**Controls**

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

Remarks

You can manipulate **Control** objects using the reference returned by the **Controls** property.

DataBinding Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjDatabindingObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjDatabindingObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjDatabindingObjectP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjDatabindingObjectM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjDatabindingObjectE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjDatabindingObjectS"}

The **DataBinding** object represents a bindable property of a component.

Syntax

DataBinding

Remarks

There is one **DataBinding** object for each property of a component marked as Bindable in the **Procedure Attributes** dialog box.

Visual Basic version 4.0 supported binding only one property of a control to a database at a time. Visual Basic 5.0, however, gives you the ability to bind multiple properties of a control to a database. This is used most commonly with **User** controls. For more information on this, see Chapter 9 in "Creating ActiveX Components" in the *Component Tools Guide*.

DefaultBind Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDefaultBindPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDefaultBindPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDefaultBindPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDefaultBindPropertyS"}

Returns or sets the DefaultBind attribute of a **Member** object.

Syntax

object.**DefaultBind**

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

DisplayBind Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDisplayBindPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDisplayBindPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDisplayBindPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDisplayBindPropertyS"}

Returns or sets the DisplayBind attribute of a **Member** object.

Syntax

object.**DisplayBind**

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

Events Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproEventsPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproEventsPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproEventsPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproEventsPropertyS"}

Supplies properties that enable add-ins to connect to all events in Visual Basic for Applications.

Syntax

object.**Events**

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

Hidden Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproHiddenPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHiddenPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHiddenPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHiddenPropertyS"}

Returns or sets the Hidden attribute of a **Member** object.

Syntax

object.**Hidden**

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

HScrollSmallChange, VScrollSmallChange Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproHScrollSmallChangePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHScrollSmallChangePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHScrollSmallChangePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHScrollSmallChangePropertyS"}

Returns or sets the distance the **UserDocument** will scroll when the user clicks a scroll arrow.

Syntax

object.**HScrollSmallChange** = *single*
object.**VScrollSmallChange** = *single*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
single	The distance in twips the **UserDocument** will scroll when the user clicks the scroll arrow.

Remarks

There is no "LargeChange" property counterpart to the **HScrollSmallChange** and **VScrollSmallChange** properties. The "LargeChange" is determined by the **ViewPort** object's **ViewPortHeight** and **ViewPortWidth** properties.

Hyperlink Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjHyperlinkObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjHyperlinkObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjHyperlinkObjectP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjHyperlinkObjectM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjHyperlinkObjectE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjHyperlinkObjectS"}

Using the properties and methods of the **Hyperlink** object, your ActiveX document or ActiveX control can request a hyperlink-aware container, such as Microsoft Internet Explorer, to jump to a given URL.

Remarks

Use the **NavigateTo** method to jump to a URL. For example, the following code presumes an ActiveX document named "axdMyDoc" exists:

```
UserDocument.Hyperlink.NavigateTo _
"c:\mydocs\axdmydoc.vbd"
```

If your ActiveX document is contained by a hyperlink-aware container (such as Internet Explorer), and if the container maintains a history of documents, use the **GoBack** or **GoForward** methods to go backwards or forwards through the list. However, be sure to use error-checking, as shown in the example below:

```
Private Sub cmdGoForward_Click()
   On Error GoTo noDocInHistory
   UserDocument.Hyperlink.GoForward
   Exit Sub
noDocInHistory:
   Resume Next
End Sub
```

Hyperlink Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproHyperlinkPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHyperlinkPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHyperlinkPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHyperlinkPropertyS"}

Returns a reference to the **Hyperlink** object.

Syntax

object.**Hyperlink**

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

ItemReloaded Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtItemReloadedEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtItemReloadedEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtItemReloadedEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtItemReloadedEventS"}

Occurs after a component is reloaded.

Syntax

Private Sub *object*_**ItemReloaded(***vbcomponent* **As VBComponent)**

The ItemReloaded event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
vbcomponent	A **VBComponent** object representing the component that was reloaded.

Lines Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproLinesPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLinesPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLinesPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLinesPropertyS"}

Returns a string containing the specified block of lines.

Syntax

object.**Lines(***startline* **As Long**, *count* **As Long)**

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
startline	A Long data type specifying the line number to start at.
count	A Long data type specifying the number of lines to highlight.

LogEvent Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproLogEventMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLogEventMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLogEventMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLogEventMethodS"}

Logs an event in the application's log target. On Windows NT platforms, the method writes to the NT Event log. On Windows 95 platforms, the method writes to the file specified in the **LogPath** property).

Syntax

object.**LogEvent (***logBuffer, eventType***)**

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
logBuffer	Required. String to be written to the log.
eventType	Optional. A Long integer that specifies the type of event, as shown in Settings.

Settings

The settings for *eventType* are:

Constant	Value	Description
EVENTLOG_ERROR_TYPE	1	Error.
EVENTLOG_WARNING_TYPE	2	Warning.
EVENTLOG_INFORMATION_TYPE	4	Information.

Remarks

Guidelines for logging are available in the Win32 SDK, and those guidelines should be followed when logging either to the NT Event log or the file specified in the **LogPath** property (on Windows 95 platforms).

LogMode Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproLogModePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLogModePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLogModePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLogModePropertyS"}

Returns or sets a value which determines how logging (through the **LogEvent** method) will be carried out.

Syntax

object.**LogMode =** *mode*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
mode	Long. Determines the method of logging, as shown in Settings below.

Settings

The settings for *mode* are:

Constant	Value	Description
vbLogAuto	0	If running on Windows 95, this option logs messages to the file specified in the **LogFile** property. If running on Windows NT, messages are logged to the NT Application Event Log, with the App.Title string used as the application source.
VbLogOff	1	Turns all logging off. Messages from UI shunts as well as from the **LogEvent** method are ignored and discarded.
VbLogToFile	2	Forces logging to a file. If no valid filename is present in **LogPath**, logging is ignored, and the property is set to **vbLogOff**.
VbLogToNT	3	Forces logging to the NT event log. If not running on Windows NT, or the event log is unavailable, logging is ignored and the property is set to **vbLogOff**.
VbLogOverwrite	0x10	Indicates that the logfile should be recreated each time the application starts. This value can be combined with other mode options using the OR operator. The default action for logging is to append to the existing file. In the case of NT event logging, this flag has no meaning.
VbLogThreadID	0x20	Indicates that the current thread ID be prepended to the message, in the form "[T:0nnn] ". This value can be combined with other mode options using the OR operator. The default action is to show the thread ID only when the application is multi-threaded (either explicitly marked as thread-safe, or implemented as an implicit multithreaded app, such as a local server with the instancing

property set to Single-Use, multithreaded).

Return Type

Long

LogPath Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproLogPathPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLogPathPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLogPathPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLogPathPropertyS"}

Returns or sets the path and filename of the file used to capture output from the **LogEvent** method. Not available at design-time.

Syntax

object.**LogPath =** *path*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
path	String. The path and filename of a log file.

Remarks

The **LogMode** property determines how logging will be carried out. If no **LogPath** is set, the **LogEvent** method writes to the NT LogEvent file.

Members Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproMembersPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMembersPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMembersPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMembersPropertyS"}

Contains identifiers that have module-level scope and can be considered properties, methods, or events of the specified **CodeModule** object.

Syntax

object.**Members**

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

MinHeight, MinWidth Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproMinHeightPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMinHeightPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMinHeightPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMinHeightPropertyS"}

Returns or sets the minimum height or width of the Viewport at which scrollbars will appear on the container.

Syntax

object.**MinHeight** = *single*
object.**MinWidth** = *single*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
single	The height or width of a UserDocument at which scrollbars will appear on a container.

Remarks

The default values of the **MinHeight** and **MinWidth** properties are set by the **Height** and **Width** properties of the **UserDocument**.

The **MinWidth** and **MinHeight** have no effect if the **ScrollBars** property is set to **False**.

Moveable Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproMoveablePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMoveablePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMoveablePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMoveablePropertyS"}

Returns or sets a value which specifies if the object can be moved.

Syntax

object.**Moveable =** *boolean*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that specifies if the object can be moved.

Settings

The settings for *boolean* are:

Constant	Value	Description
True	-1	The object can be moved.
False	0	The object cannot be moved.

NonModalAllowed Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproNonModalAllowedPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproNonModalAllowedPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproNonModalAllowedPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproNonModalAllowedPropertyS"}

Returns a value which indicates if a form can be shown non-modally (modeless). Not available at design-time.

Syntax

object.**nonModalAllowed**

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

Return Type

Boolean

Palette Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproPalettePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPalettePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPalettePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPalettePropertyS"}

Returns or sets an image that contains the palette to use for the control.

Syntax

object.**Palette =** *path*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
path	The path of the bitmap image containing the palette to be used.

Remarks

You can use a .dib file to set the palette as well as .bmp files.

PaletteMode Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproPaletteModePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPaletteModePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPaletteModePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPaletteModePropertyS"}

Returns or sets a value that determines which palette to use for the controls on a object.

Syntax

object.**PaletteMode =** *integer*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
integer	Determines the palette mode to be used, as described in Settings, below.

Settings

The settings for *integer* are:

Constant	Value	Description
vbPaletteModeHalfTone	0	(Default) Use the Halftone palette.
vbPaletteModeUseZOrder	1	Use the palette from the topmost control that has a palette.
vbPaletteModeCustom	2	Use the palette specified in the **Palette** property.
vbPaletteModeContainer	3	Use the container's palette for containers that support ambient **Palette** property. Applies to UserControls only.
vbPaletteModeNone	4	Do not use any palette. Applies to UserControls only.
vbPaletteModeObject	5	Use the ActiveX designer's palette. (Applies only to ActiveX designers which contain a palette.)

Remarks

If no palette is available, the halftone palette becomes the default palette.

PropertyName Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproPropertyNamePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPropertyNamePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPropertyNamePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPropertyNamePropertyS"}

The behavior of the **PropertyName** property depends upon the context in which it is being used.

- **AsyncRead** method — Sets the name of the property that will be associated with the **AsyncProperty** object's **Value** property.
- AsyncReadComplete event — Specifies the name of the property currently being read. This should correspond to a name assigned to the **AsyncProperty** object when invoking the **AsyncRead** method.

Syntax

object.**PropertyName** = *string*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	The name of a property to be saved or retrieved.

PropertyName Property Example

The example assigns a value to the **PropertyName** property in the **AsyncRead** method. The same value will be used to assign the result of the method to a **PictureBox** control. To try the example, place a **PictureBox** control on a **UserDocument** object. Paste the code into the General section, and press F5 to run. Start Internet Explorer 3.0 (or later), and type the path and file name of the UserDocument.vbd file into the **Address** box.

```
Private Sub UserDocument_InitProperties()
   Dim strPath As String
   ' Set the variable to a valid path for a bitmap
   ' on your computer.
   strPath = "C:\Program Files\DevStudio\VB\" & _
      "Samples\PGuide\VCR\Bfly1.bmp"
   AsyncRead strPath, vbAsyncTypeFile, _
      PropertyName:= "butterfly"
End Sub

Private Sub UserDocument_AsyncReadComplete (AsyncProp _
   As AsyncProperty)
   ' Use the Select statement to determine which
   ' Property is being returned.
   Select Case AsyncProp.PropertyName
      Case "butterfly"
         Picture1.Picture = _
         LoadPicture(AsyncProp.Value)
   End Select
End Sub
```

PropertyPage Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjPropertyPageObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjPropertyPageObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjPropertyPageObjectP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjPropertyPageObjectM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjPropertyPageObjectE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjPropertyPageObjectS"}

The base object used to create an ActiveX Property Page.

Remarks

Property pages provide an alternative to the Properties window for viewing properties. You can group several related properties on a page, or use a page to provide a dialog-like interface for a property that's too complex for the Properties window. A **PropertyPage** object represents one page, which is to say one tab in the **Property Pages** dialog box.

PropertyPage Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproPropertyPagePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPropertyPagePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPropertyPagePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPropertyPagePropertyS"}

Returns or sets the PropertyPage attribute of a **Member** object.

Syntax

object.**PropertyPage**

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

Quit Method (Add-Ins)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthQuitMethodAddInsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthQuitMethodAddInsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthQuitMethodAddInsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthQuitMethodAddInsS"}

Attempts to exit Visual Basic.

Syntax

object.**Quit**

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

RemoveAddInFromToolbar Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthRemoveAddInFromToolbarMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthRemoveAddInFromToolbarMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthRemoveAddInFromToolbarMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthRemoveAddInFromToolbarMethodS"}

Removes a button from the Add-In toolbar which references an add-in or Wizard.

Syntax

object.**RemoveAddInFromToolbar** (*saddinname* **As String**)

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
saddinname	Required. A string expression specifying the name of the add-in or Wizard to remove from the Add-In toolbar (as specified by the *saddinname* parameter from the **AddToAddInToolbar** method).

RemoveAddInFromToolbar Method Example

This example removes an existing button from the Add-In toolbar that references a fictitious add-in called MyAddIn Title:

```
Sub Main()
  dim x as Object
  Set x=CreateObject("AddInToolbar.Manager")
  x.RemoveAddInFromToolbar sAddInName:="MyAddIn Title"
End Sub
```

RequestEdit Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproRequestEditPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRequestEditPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRequestEditPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRequestEditPropertyS"}

Returns or sets the RequestEdit attribute of a **Member** object.

Syntax

object.**RequestEdit**

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

SelectAll Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthSelectAllMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthSelectAllMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthSelectAllMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthSelectAllMethodS"}

Selects all of the controls contained on a form.

Syntax

object.**SelectAll**

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

SetViewport Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthSetViewPortMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthSetViewPortMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthSetViewPortMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthSetViewPortMethodS"}

Sets the left and top coordinates of the **UserDocument** that will be visible in the Viewport.

Syntax

object.**SetViewPort** *left, top*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
left	Required. A value of type Single that specifies the left coordinate of the UserDocument.
top	Required. A value of type Single that specifies the top coordinate of the UserDocument.

SetViewport Method Example

The example uses **SetViewport** method to automatically place the **TextBox** control with focus into the top left corner of the Viewport of container. To try the example, place an array of three or more **TextBox** controls onto a **UserDocument** object. Paste the code below into the General section. Press F5 to run the project, then run Internet Explorer (3.0 or later). In Internet Explorer, type the path and file name of the ActiveX document (UserDocument1.vbd) into the **Address** box (the file will be in the same directory as the Visual Basic executable). When the ActiveX document is displayed, type any distinctive text into the first **TextBox** control. Press TAB to move to the next control to see the effect of the **SetViewPort** method.

```
Private Sub Text1_GotFocus(Index As Integer)
   UserDocument.SetViewport Text1(Index).Left, _
      Text1(Index).Top
End Sub

Private Sub UserDocument_Initialize()
   ' The container must be small enough for scrollbars
   ' to appear. To assure this, set the MinHeight and
   ' MinWidth properties to be larger than the
   ' container.
   UserDocument.MinHeight = 10000
   UserDocument.MinWidth = 10000
End Sub
```

Size Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthSizeMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthSizeMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthSizeMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthSizeMethodS"}

Changes the width and height of a **UserControl** object.

Syntax

object.**Size** *width, height*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
width	Required. The width in twips of the object.
height	Required. The height in twips of the object.

Remarks

The **Width** and **Height** properties of a **UserControl** object are always given in Twips, regardless of **ScaleMode**.

StandardMethod Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproStandardMethodPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproStandardMethodPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproStandardMethodPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproStandardMethodPropertyS"}

Returns or sets the StandardMethod attribute of a **Member** object.

Syntax

object.**StandardMethod**

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

StartLogging Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproStartLoggingMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproStartLoggingMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproStartLoggingMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproStartLoggingMethodS"}

Sets the log target and log mode of an operation.

Syntax

object.**StartLogging** *logTarget, logMode*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
logTarget	Path and filename of the file used to capture output from the **LogEvent** method.
logMode	A value which determines how logging (through the **LogEvent** method) will be carried out. See Settings below.

Settings

The settings for *logMode* are:

Constant	Value	Description
vbLogAuto	0	If running on Windows 95, this option logs messages to the file specified in the **LogFile** property. If running on Windows NT, messages are logged to the NT Application Event Log, with the App.Title string used as the application source.
VbLogOff	1	Turns all logging off. Messages from UI shunts as well as from the **LogEvent** method are ignored and discarded.
VbLogToFile	2	Forces logging to a file. If no valid filename is present in **LogPath**, logging is ignored, and the property is set to **vbLogOff**.
VbLogToNT	3	Forces logging to the NT event log. If not running on Windows NT, or the event log is unavailable, logging is ignored and the property is set to **vbLogOff**.
VbLogOverwrite	0x10	Indicates that the logfile should be recreated each time the application starts. This value can be combined with other mode options using the **OR** operator. The default action for logging is to append to the existing file. In the case of NT event logging, this flag has no meaning.
VbLogThreadID	0x20	Indicates that the current thread ID be prepended to the message, in the form "[T:0nnn] ". This value can be combined with other mode options using the **OR** operator. The default action is to show the thread ID only when the application is multi-threaded (either explicitly marked as thread-safe, or

implemented as an implicit multithreaded app, such as a local server with the instancing property set to Single-Use, multithreaded).

ThreadID Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproTheadIDPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTheadIDPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTheadIDPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTheadIDPropertyS"}

Returns the Win32 ID of the executing thread. (Used for Win32 API calls.)

Syntax

object.**ThreadID**

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

Return Type

Long

UIDefault Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproUIDefaultPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproUIDefaultPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproUIDefaultPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproUIDefaultPropertyS"}

Returns or sets the UIDefault attribute of a **Member** object.

Syntax

object.**UIDefault**

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

UnattendedApp Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproUnattendedAppPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproUnattendedAppPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproUnattendedAppPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproUnattendedAppPropertyS"}

Returns or sets a value that determines if an application will run without any user interface.

Syntax

object.**UnattendedApp=** *boolean*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that specifies if the application will run without any user interface.

Settings

The settings for *boolean* are:

Constant	Value	Description
True	-1	The application has no user interface.
False	0	The application has a user interface.

UserControl Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjUserControlObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjUserControlObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjUserControlObjectP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjUserControlObjectM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjUserControlObjectE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjUserControlObjectS"}

The **UserControl** object is the base object used to create an ActiveX control.

Remarks

An ActiveX control created with Visual Basic is always composed of a **UserControl** object, plus any controls — referred to as constituent controls — that you choose to place on the **UserControl**.

Like Visual Basic forms, **UserControl** objects have code modules and visual designers. Place constituent controls on the **UserControl** object's designer, just as you would place controls on a form.

UserDocument Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjUserDocumentObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjUserDocumentObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjUserDocumentObjectP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjUserDocumentObjectM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjUserDocumentObjectE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjUserDocumentObjectS"}

The base object of an ActiveX document, the **UserDocument** object resembles a standard Visual Basic **Form** object with some exceptions.

Remarks

The **UserDocument** object has most, but not all, of the events that are found on a **Form** object. The events present on a **Form** that are not found on the **UserDocument** include: Activate, Deactivate, LinkClose, LinkError, LinkExecute, LinkOpen, Load, QueryUnload, and Unload events.

Events present on the **UserDocument**, but not found on a **Form** object include: AsycReadComplete, EnterFocus, ExitFocus, Hide, InitProperties, ReadProperties, Scroll, Show, and WriteProperties events.

You cannot place embedded objects (such as an Excel or Word document) or an **OLE Container** control on a **UserDocument**.

VBProjects Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproVBProjectsPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproVBProjectsPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproVBProjectsPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproVBProjectsPropertyS"}

Returns the VBProjects collection, which represents all of the projects currently open in the Visual Basic IDE.

Syntax

object.**VBProjects**

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

ViewportHeight, ViewportLeft, ViewportTop, ViewportWidth Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproViewPortHeightPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproViewPortHeightPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproViewPortHeightPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproViewPortHeightPropertyS"}

Returns the current height, left, top, or width value of the Viewport.

Syntax

object.**ViewportHeight**
object.**ViewportLeft**
object.**ViewportTop**
object.**ViewportWidth**

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

Return Type

Single

Remarks

The application used to view the ActiveX document controls the size of the Viewport. However, you can use the **MinHeight** and **MinWidth** properties to resize the **UserDocument**. For example, the code below resizes a **PictureBox** control according to the size of the Viewport left height and width properties.

```
Private Sub UserDocument_Resize()
   Picture1.Width = UserDocument.ViewportWidth - _
      Picture1.Left
   Picture1.Height = UserDocument.ViewportHeight - _
      Picture1.Top
End Sub
```

Windows Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproWindowsPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproWindowsPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproWindowsPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproWindowsPropertyS"}

Returns the **Window** object, which represents a window in the Visual Basic IDE.

Syntax

object.**Window**

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

Charset Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproCharsetPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCharsetPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCharsetPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCharsetPropertyS"}

Sets or returns the character set used in the font.

Syntax

object.**Charset** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A integer value that specifies the character set used by the font, as described in Settings.

Settings

The following are some common settings for *value*:

Value	Description
0	Standard Windows characters
2	The symbol character set.
128	Double-byte character set (DBCS) unique to the Japanese version of Windows
255	Extended characters normally displayed by DOS applications.

Remarks

Setting the **Charset** property to one of its available values selects the character set only if it is available in the current font.

DataBindings Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDataBindingsPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDataBindingsPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDataBindingsPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDataBindingsPropertyS"}

Returns the **DataBindings** collection object containing the bindable properties available to the developer.

Syntax

object.**DataBindings**

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

Control Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjControlObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjControlObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjControlObjectP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjControlObjectM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjControlObjectE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjControlObjectS"}

The class name of all Visual Basic internal controls.

Syntax

Control

Remarks

You can dimension a variable as a **Control** object and reference it as you would a control on a form. The following demonstrates this:

```
Dim C as Control
Set C = Command1
```

IsBindable Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproIsBindablePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproIsBindablePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproIsBindablePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproIsBindablePropertyS"}

Returns a boolean value indicating whether the property is bindable. This property is read-only.

Syntax

object.**IsBindable**

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

Remarks

Use this property to determine if the property is bindable.

Note This property is usually used in a Wizard to check whether a property is bindable.

IsDataSource Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproIsDataSourcePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproIsDataSourcePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproIsDataSourcePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproIsDataSourcePropertyS"}

Returns a boolean value indicating whether the property is a data source. This property is read-only.

Syntax

object.**IsDataSource**

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

Remarks

Use this property to determine if the property is a data source and can be attached to a data control.

Note This property is usually used in a Wizard to check whether a property is a data source.

UseMaskColor Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproUseMaskColorPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproUseMaskColorPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproUseMaskColorPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproUseMaskColorPropertyS"}

Returns or sets a value that determines whether the color assigned in the **MaskColor** property is used as a mask.

Syntax

object.**UseMaskColor** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A boolean value that determines whether the **MaskColor** property is used, as described in Settings.

Settings

The settings for *value* are:

Value	Description
True	The **MaskColor** is used to create transparent regions.
False	(Default) The **MaskColor** is not used.

Remarks

The **MaskColor** and **UseMaskColor** properties are used only when the **Style** property is set to 1 - Graphical.

RightToLeft Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproRightToLeftPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRightToLeftPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRightToLeftPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRightToLeftPropertyS"}

Determines the text input method on a bidirectional system.

Syntax

object.**RightToLeft** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A boolean value that determines the input method on a bidirectional system, as described in Settings.

Settings

The settings for *value* are:

Value	Description
True	Text is entered from right to left.
False	(Default) Text is entered from left to right.

Remarks

The **RightToLeft** property is only available on bidirectional systems. Setting this property on a non bidirectional system has no effect.

Resync Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthResyncMethodRemoteDataC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthResyncMethodRemoteDataX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthResyncMethodRemoteDataA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthResyncMethodRemoteDataS"}

Fetches the batch conflict values for the current row.

Syntax

object.**Resync**

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

Remarks

The **Resync** method is valid only when using Client-Batch Cursors.

Resync resynchronizes the columns in the current row in the cursor library with the current data on the server (visible to your transaction). If you have not modified the row, this method changes the **Value** and **OriginalValue** properties to match what is currently on the server.

If you have modified the row, this method will only adjust the **OriginalValue** property so as not to loose your edits. This second case is useful when you want to avoid an optimistic concurrency conflict.

The last case where this is used is when you're dealing with a row that you attempted to update using **BatchUpdate**, but a conflict occurred because the concurrency check failed. In this case, this method will adjust the **BatchConflictValue** to reflect the most recent version of the column on the server.

Visual Basic Constants

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

The following constants are specified or recognized by Visual Basic. As a result, they can be used anywhere in your code in place of the actual values.

- Alignment Constants
- **Border** Property Constants
- **Clipboard** Object Constants
- Color Constants
- **CommonDialog** Control Constants
- **CommonDialog** Error Constants
- Control Constants
- **Data** Control Constants
- **DBGrid** Control Constants
- **DBList** and **DBCombo** Controls Constants
- DDE Constants
- Drag-and-Drop Constants
- Drawing Constants
- Form Constants
- Graphics Constants
- **Grid** Control Constants
- Help Constants
- Key Code Constants
- Menu Accelerator Constants
- **Menu** Control Constants
- Miscellaneous Constants
- Mouse Pointer Constants
- **OLE** Container Control Constants
- **Picture** Object Constants
- **Printer** Object Constants
- RasterOp Constants
- Variant Type Constants

Use the Object Browser to view the intrinsic constants you can use with methods and properties. From the View menu, choose Object Browser, select the Visual Basic object library, and then choose the Constants object. You can scroll through the constants that appear under Methods/Properties.

Miscellaneous Constants

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

ZOrder Method

Constant	Value	Description
vbBringToFront	0	Bring to front
vbSendToBack	1	Send to back

QueryUnload Method

Constant	Value	Description
vbAppWindows	2	Current Windows session ending
vbFormMDIForm	4	MDI child form is closing because the MDI form is closing
vbFormCode	1	**Unload** method invoked from code
vbFormControlMenu	0	User has chosen Close command from the Control-menu box on a form
vbAppTaskManager	3	Windows Task Manager is closing the application

Shift Parameter Masks

Constant	Value	Description
vbShiftMask	1	SHIFT key bit mask
vbCtrlMask	2	CTRL key bit mask
vbAltMask	4	ALT key bit mask

Mouse Button Parameter Masks

Constant	Value	Description
vbLeftButton	1	Left mouse button
vbRightButton	2	Right mouse button
vbMiddleButton	4	Middle mouse button

Application Start Mode

Constant	Value	Description
vbSModeStandalone	0	Stand-alone application
vbSModeAutomation	1	OLE automation server

LoadResPicture Method

Constant	Value	Description
vbResBitmap	0	Bitmap resource
vbResIcon	1	Icon resource
vbResCursor	2	Cursor resource

Control Constants

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

ComboBox Control

Constant	Value	Description
vbComboDropdown	0	Dropdown Combo
vbComboSimple	1	Simple Combo
vbComboDropdownList	2	Dropdown List

CheckBox Control

Constant	Value	Description

vbUnchecked	0	Unchecked
vbChecked	1	Checked
vbGrayed	2	Grayed

ListBox Control

Constant	Value	Description
vbMultiSelectNone	0	None
vbMultiSelectSimple	1	Simple
vbMultiSelectExtended	2	Extended

ScrollBar Control

Constant	Value	Description
vbSBNone	0	None
vbHorizontal	1	Horizontal
vbVertical	2	Vertical
vbBoth	3	Both

Shape Control

Constant	Value	Description
vbShapeRectangle	0	Rectangle
vbShapeSquare	1	Square
vbShapeOval	2	Oval
vbShapeCircle	3	Circle
vbShapeRoundedRectangle	4	Rounded rectangle
vbShapeRoundedSquare	5	Rounded square

Grid Control Constants

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

ColAlignment, FixedAlignment Properties

Constant	Value	Description
grdAlignCenter	2	Center data in column
grdAlignLeft	0	Left-align data in column
grdAlignRight	1	Right-align data in column

FillStyle Property

Constant	Value	Description
grdSingle	0	Changing **Text** property setting affects only active cell
grdRepeat	1	Changing **Text** property setting affects all selected cells

Picture Object Constants

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

Constant	Value	Description
vbPicTypeNone	0	Icon type of **Picture** object
vbPicTypeBitmap	1	Bitmap type of **Picture** object
vbPicTypeMetafile	2	Metafile type of **Picture** object
vbPicTypeIcon	3	Icon type of **Picture** object
vbPicTypeEMetaFile	4	Enhanced metafile type of **Picture** object

Variant Type Constants

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

Constant	Value	Description
vbVEmpty	0	Empty (uninitialized)
vbVNull	1	Null (no valid data)
vbVInteger	2	Integer data type
vbVLong	3	Long integer data type
vbVSingle	4	Single-precision floating-point data type
vbVDouble	5	Double-precision floating-point data type
vbVCurrency	6	Currency (scaled integer) data type
vbVDate	7	Date data type
vbVString	8	String data type

DDE Constants

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

linkerr (LinkError Event)

Constant	Value	Description
vbWrongFormat	1	Another application requested data in wrong format
vbDDESourceClosed	6	Destination application attempted to continue after source closed
vbTooManyLinks	7	All source links are in use
vbDataTransferFailed	8	Failure to update data in destination

LinkMode Property (Forms and Controls)

Constant	Value	Description
vbLinkNone	0	None
vbLinkSource	1	Source (forms only)
vbLinkAutomatic	1	Automatic (controls only)
vbLinkManual	2	Manual (controls only)
vbLinkNotify	3	Notify (controls only)

LinkMode Property (Only for backward compatibility with Visual Basic version 1.0; use new constants instead)

Constant	Value	Description
vbHot	1	Hot (controls only)
vbServer	1	Server (forms only)
vbCold	2	Cold (controls only)

Clipboard Object Constants

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

Constant	Value	Description
vbCFLink	&HBF00	DDE conversation information
vbCFRTF	&HBF01	Rich Text Format (.rtf file)
vbCFText	1	Text (.txt file)
vbCFBitmap	2	Bitmap (.bmp file)
vbCFMetafile	3	Metafile (.wmf file)
vbCFDIB	8	Device-independent bitmap
vbCFPalette	9	Color palette

Drag-and-Drop Constants

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

DragOver Event

Constant	Value	Description
vbEnter	0	Source control dragged into target
vbLeave	1	Source control dragged out of target
vbOver	2	Source control dragged from one position in target to another

Drag Method (Controls)

Constant	Value	Description
vbCancel	0	Cancel drag operation
vbBeginDrag	1	Begin dragging control
vbEndDrag	2	Drop control

DragMode Property

Constant	Value	Description
vbManual	0	Manual
vbAutomatic	1	Automatic

Form Constants

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

Show Parameters

Constant	Value	Description
vbModal	1	Modal form
vbModeless	0	Modeless form

Arrange Method for MDI Forms

Constant	Value	Description
vbCascade	0	Cascade all nonminimized MDI child forms
vbTileHorizontal	1	Horizontally tile all nonminimized MDI child forms
vbTileVertical	2	Vertically tile all nonminimized MDI child forms
vbArrangeIcons	3	Arrange icons for minimized MDI child forms

WindowState Property

Constant	Value	Description
vbNormal	0	Normal
vbMinimized	1	Minimized
vbMaximized	2	Maximized

Key Code Constants

Key Codes

Constant	Value	Description
vbKeyLButton	&H1	Left mouse button
vbKeyRButton	&H2	Right mouse button
vbKeyCancel	&H3	CANCEL key
vbKeyMButton	&H4	Middle mouse button
vbKeyBack	&H8	BACKSPACE key
vbKeyTab	&H9	TAB key
vbKeyClear	&HC	CLEAR key
vbKeyReturn	&HD	ENTER key
vbKeyShift	&H10	SHIFT key
vbKeyControl	&H11	CTRL key
vbKeyMenu	&H12	MENU key
vbKeyPause	&H13	PAUSE key
vbKeyCapital	&H14	CAPS LOCK key
vbKeyEscape	&H1B	ESC key
vbKeySpace	&H20	SPACEBAR key
vbKeyPageUp	&H21	PAGE UP key
vbKeyPageDown	&H22	PAGE DOWN key
vbKeyEnd	&H23	END key
vbKeyHome	&H24	HOME key
vbKeyLeft	&H25	LEFT ARROW key
vbKeyUp	&H26	UP ARROW key
vbKeyRight	&H27	RIGHT ARROW key
vbKeyDown	&H28	DOWN ARROW key
vbKeySelect	&H29	SELECT key
vbKeyPrint	&H2A	PRINT SCREEN key
vbKeyExecute	&H2B	EXECUTE key
vbKeySnapshot	&H2C	SNAPSHOT key
vbKeyInsert	&H2D	INS key
vbKeyDelete	&H2E	DEL key
vbKeyHelp	&H2F	HELP key
vbKeyNumlock	&H90	NUM LOCK key

KeyA Through KeyZ Are the Same as Their ASCII Equivalents: 'A' Through 'Z'

Constant	Value	Description
vbKeyA	65	A key
vbKeyB	66	B key
vbKeyC	67	C key
vbKeyD	68	D key
vbKeyE	69	E key

vbKeyF	70	F key
vbKeyG	71	G key
vbKeyH	72	H key
vbKeyI	73	I key
vbKeyJ	74	J key
vbKeyK	75	K key
vbKeyL	76	L key
vbKeyM	77	M key
vbKeyN	78	N key
vbKeyO	79	O key
vbKeyP	80	P key
vbKeyQ	81	Q key
vbKeyR	82	R key
vbKeyS	83	S key
vbKeyT	84	T key
vbKeyU	85	U key
vbKeyV	86	V key
vbKeyW	87	W key
vbKeyX	88	X key
vbKeyY	89	Y key
vbKeyZ	90	Z key

Key0 Through Key9 Are the Same as Their ASCII Equivalents: '0' Through '9

Constant	Value	Description
vbKey0	48	0 key
vbKey1	49	1 key
vbKey2	50	2 key
vbKey3	51	3 key
vbKey4	52	4 key
vbKey5	53	5 key
vbKey6	54	6 key
vbKey7	55	7 key
vbKey8	56	8 key
vbKey9	57	9 key

Keys on the Numeric Keypad

Constant	Value	Description
vbKeyNumpad0	&H60	0 key
vbKeyNumpad1	&H61	1 key
vbKeyNumpad2	&H62	2 key
vbKeyNumpad3	&H63	3 key
vbKeyNumpad4	&H64	4 key
vbKeyNumpad5	&H65	5 key
vbKeyNumpad6	&H66	6 key
vbKeyNumpad7	&H67	7 key

vbKeyNumpad8	&H68	8 key
vbKeyNumpad9	&H69	9 key
vbKeyMultiply	&H6A	MULTIPLICATION SIGN (*) key
vbKeyAdd	&H6B	PLUS SIGN (+) key
vbKeySeparator	&H6C	ENTER (keypad) key
vbKeySubtract	&H6D	MINUS SIGN (-) key
vbKeyDecimal	&H6E	DECIMAL POINT(.) key
vbKeyDivide	&H6F	DIVISION SIGN (/) key

Function Keys

Constant	Value	Description
vbKeyF1	&H70	F1 key
vbKeyF2	&H71	F2 key
vbKeyF3	&H72	F3 key
vbKeyF4	&H73	F4 key
vbKeyF5	&H74	F5 key
vbKeyF6	&H75	F6 key
vbKeyF7	&H76	F7 key
vbKeyF8	&H77	F8 key
vbKeyF9	&H78	F9 key
vbKeyF10	&H79	F10 key
vbKeyF11	&H7A	F11 key
vbKeyF12	&H7B	F12 key
vbKeyF13	&H7C	F13 key
vbKeyF14	&H7D	F14 key
vbKeyF15	&H7E	F15 key
vbKeyF16	&H7F	F16 key

Color Constants

Colors

Constant	Value	Description
vbBlack	&H0	Black
vbRed	&HFF	Red
vbGreen	&HFF00	Green
vbYellow	&HFFFF	Yellow
vbBlue	&HFF0000	Blue
vbMagenta	&HFF00FF	Magenta
vbCyan	&HFFFF00	Cyan
vbWhite	&HFFFFFF	White

System Colors

Constant	Value	Description
vbScrollBars	&H80000000	Scroll bar color
vbDesktop	&H80000001	Desktop color
vbActiveTitleBar	&H80000002	Color of the title bar for the active window
vbInactiveTitleBar	&H80000003	Color of the title bar for the inactive window
vbMenuBar	&H80000004	Menu background color
vbWindowBackground	&H80000005	Window background color
vbWindowFrame	&H80000006	Window frame color
vbMenuText	&H80000007	Color of text on menus
vbWindowText	&H80000008	Color of text in windows
vbTitleBarText	&H80000009	Color of text in caption, size box, and scroll arrow
vbActiveBorder	&H8000000A	Border color of active window
vbInactiveBorder	&H8000000B	Border color of inactive window
vbApplicationWorkspace	&H8000000C	Background color of multiple-document interface (MDI) applications
vbHighlight	&H8000000D	Background color of items selected in a control
vbHighlightText	&H8000000E	Text color of items selected in a control
vbButtonFace	&H8000000F	Color of shading on the face of command buttons
vbButtonShadow	&H80000010	Color of shading on the

		edge of command buttons
vbGrayText	&H80000011	Grayed (disabled) text
vbButtonText	&H80000012	Text color on push buttons
vbInactiveCaptionText	&H80000013	Color of text in an inactive caption
vb3DHighlight	&H80000014	Highlight color for 3D display elements
vb3DDKShadow	&H80000015	Darkest shadow color for 3D display elements
vb3DLight	&H80000016	Second lightest of the 3D colors after **vb3Dhighlight**
vb3DFace	&H8000000F	Color of text face
vb3DShadow	&H80000010	Color of text shadow
vbInfoText	&H80000017	Color of text in ToolTips
vbInfoBackground	&H80000018	Background color of ToolTips

Alignment Constants

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

Align Property

Constant	Value	Description
vbAlignNone	0	Size and location set at design time or in code
vbAlignTop	1	Align control to top of form
vbAlignBottom	2	Align control to bottom of form
vbAlignLeft	3	Align control to left of form
vbAlignRight	4	Align control to right of form

Alignment Property

Constant	Value	Description
vbLeftJustify	0	Left align
vbRightJustify	1	Right align
vbCenter	2	Center

Border Property Constants

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

BorderStyle Property (Form)

Constant	Value	Description
vbBSNone	0	No border
vbFixedSingle	1	Fixed single
vbSizable	2	Sizable (forms only)
vbFixedDouble	3	Fixed double (forms only)

BorderStyle Property (Shape and Line)

Constant	Value	Description
vbTransparent	0	Transparent
vbBSSolid	1	Solid
vbBSDash	2	Dash
vbBSDot	3	Dot
vbBSDashDot	4	Dash-dot
vbBSDashDotDot	5	Dash-dot-dot
vbBSInsideSolid	6	Inside solid

Mouse Pointer Constants

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

Constant	Value	Description
vbDefault	0	Default
vbArrow	1	Arrow
vbCrosshair	2	Cross
vbIbeam	3	I beam
vbIconPointer	4	Icon
vbSizePointer	5	Size
vbSizeNESW	6	Size NE, SW
vbSizeNS	7	Size N, S
vbSizeNWSE	8	Size NW, SE
vbSizeWE	9	Size W, E
vbUpArrow	10	Up arrow
vbHourglass	11	Hourglass
vbNoDrop	12	No drop
vbArrowHourglass	13	Arrow and hourglass; (available only in 32-bit Visual Basic 5.0)
vbArrowQuestion	14	Arrow and question mark; (available only in 32-bit Visual Basic 5.0)
vbSizeAll	15	Size all; (available only in 32-bit Visual Basic 5.0)
vbCustom	99	Custom icon specified by the **MouseIcon** property

Drawing Constants

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

DrawMode Property

Constant	Value	Description
vbBlackness	1	Black
vbNotMergePen	2	Not Merge pen
vbMaskNotPen	3	Mask Not pen
vbNotCopyPen	4	Not Copy pen
vbMaskPenNot	5	Mask pen Not
vbInvert	6	Invert
vbXorPen	7	Xor pen
vbNotMaskPen	8	Not Mask pen
vbMaskPen	9	Mask pen
vbNotXorPen	10	Not Xor pen
vbNop	11	No operation; output remains unchanged
vbMergeNotPen	12	Merge Not pen
vbCopyPen	13	Copy pen
vbMergePenNot	14	Merge pen Not
vbMergePen	15	Merge pen
vbWhiteness	16	White

DrawStyle Property

Constant	Value	Description
vbSolid	0	Solid
vbDash	1	Dash
vbDot	2	Dot
vbDashDot	3	Dash-dot
vbDashDotDot	4	Dash-dot-dot
vbInvisible	5	Invisible
vbInsideSolid	6	Inside solid

Graphics Constants

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

FillStyle Property

Constant	Value	Description
vbFSSolid	0	Solid
vbFSTransparent	1	Transparent
vbHorizontalLine	2	Horizontal line
vbVerticalLine	3	Vertical line
vbUpwardDiagonal	4	Upward diagonal
vbDownwardDiagonal	5	Downward diagonal
vbCross	6	Cross
vbDiagonalCross	7	Diagonal cross

ScaleMode Property

Constant	Value	Description
vbUser	0	User
vbTwips	1	Twips
vbPoints	2	Points
vbPixels	3	Pixels
vbCharacters	4	Characters
vbInches	5	Inches
vbMillimeters	6	Millimeters
vbCentimeters	7	Centimeters

OLE Container Control Constants

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

OLEType Property

Constant	Value	Description
vbOLELinked	0	**OLE** container control contains a linked object
vbOLEEmbedded	1	**OLE** container control contains an embedded object
vbOLENone	3	**OLE** container control doesn't contain an object

OLETypeAllowed Property

Constant	Value	Description
vbOLEEither	2	**OLE** container control can contain either a linked or an embedded object

UpdateOptions Property

Constant	Value	Description
vbOLEAutomatic	0	Object is updated each time the linked data changes
vbOLEFrozen	1	Object is updated whenever the user saves the linked document from within the application in which it was created
vbOLEManual	2	Object is updated only when the **Action** property is set to 6 (Update)

AutoActivate Property

Constant	Value	Description
vbOLEActivateManual	0	OLE object isn't automatically activated
vbOLEActivateGetFocus	1	Object is activated when the **OLE** container control gets the focus
vbOLEActivateDoubleclick	2	Object is activated when the **OLE** container control is double-clicked
vbOLEActivateAuto	3	Object is activated based on the object's default method of activation

SizeMode Property

Constant	Value	Description
vbOLESizeClip	0	Object's image is clipped by the **OLE** container control's borders
vbOLESizeStretch	1	Object's image is sized to fill the **OLE** container control
vbOLESizeAutoSize	2	**OLE** container control is automatically

		resized to display the entire object
vbOLESizeZoom	3	Object's image is stretched but in proportion

DisplayType Property

Constant	Value	Description
vbOLEDisplayContent	0	Object's data is displayed in the **OLE** container control
vbOLEDisplayIcon	1	Object's icon is displayed in the **OLE** container control

Updated Event Constants

Constant	Value	Description
vbOLEChanged	0	Object's data has changed
vbOLESaved	1	Object's data has been saved by the application that created the object
vbOLEClosed	2	Application file containing the linked object's data has been closed
vbOLERenamed	3	Application file containing the linked object's data has been renamed

Special Verb Values

Constant	Value	Description
vbOLEPrimary	0	Default action for the object
vbOLEShow	-1	Activates the object for editing
vbOLEOpen	-2	Opens the object in a separate application window
vbOLEHide	-3	For embedded objects, hides the application that created the object
vbOLEUIActivate	-4	All UI's associated with the object are visible and ready for use
vbOLEInPlaceActivate	-5	Object is ready for the user to click inside it and start working with it
vbOLEDiscardUndoState	-6	For discarding all record of changes that the object's application can undo

Verb Flag Bit Masks

Constant	Value	Description
vbOLEFlagGrayed	&H1	Grayed menu item
vbOLEFlagDisabled	&H2	Disabled menu item
vbOLEFlagChecked	&H8	Checked menu item
vbOLEFlagSeparator	&H800	Separator bar in menu item list
vbOLEMiscFlagMemStorage	&H1	Causes control to use memory to store the object while it's loaded
vbOLEMiscFlagDisableInPlace	&H2	Forces **OLE** container control to activate objects in a separate

window

CommonDialog Control Constants

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

File Open/Save Dialog Box Flags

Constant	Value	Description
cdlOFNAllowMultiselect	&H200	Specifies that the File Name list box allows multiple selections. The user can select more than one file at run time by pressing the SHIFT key and using the UP ARROW and DOWN ARROW keys to select the desired files. When this is done, the **FileName** property returns a string containing the names of all selected files. The names in the string are delimited by spaces.
cdlOFNCreatePrompt	&H2000	Specifies that the dialog box prompts the user to create a file that doesn't currently exist. This flag automatically sets the **cdlOFNPathMustExist** and **cdlOFNFileMustExist** flags.
cdlOFNExplorer	&H80000	Use the Explorer-like Open A File dialog box template. Common dialogs that use this flag do not work under Windows NT using the Windows 95 shell.
cdlOFNExtensionDifferent	&H400	Indicates that the extension of the returned filename is different from the extension specified by the **DefaultExt** property. This flag isn't set if the **DefaultExt** property is **Null**, if the extensions match, or if the file has no extension. This flag value can be checked upon closing the dialog box.
cdlOFNFileMustExist	&H1000	Specifies that the user can enter only names of existing files in the File Name text box. If this flag is set and the user enters an invalid filename, a warning is displayed. This flag automatically sets the **cdlOFNPathMustExist** flag.
cdlOFNHelpButton	&H10	Causes the dialog box to display the Help button.
cdlOFNHideReadOnly	&H4	Hides the Read Only check box.
cdlOFNLongNames	&H200000	Use long filenames.
cdlOFNNoChangeDir	&H8	Forces the dialog box to set the current directory to what it was when the dialog box was opened.
cdlOFNNoDereferenceLinks	&H100000	Do not dereference shell links (also known as shortcuts). By default, choosing a shell link causes it to be dereferenced by the shell.
cdlOFNNoReadOnlyReturn	&H8000	Specifies that the returned file won't have the Read Only attribute set and won't be in a write-protected directory.
cdlOFNNoValidate	&H100	Specifies that the common dialog box allows invalid characters in the returned filename.

cdlOFNOverwritePrompt	&H2	Causes the Save As dialog box to generate a message box if the selected file already exists. The user must confirm whether to overwrite the file.
cdlOFNPathMustExist	&H800	Specifies that the user can enter only valid paths. If this flag is set and the user enters an invalid path, a warning message is displayed.
cdlOFNReadOnly	&H1	Causes the Read Only check box to be initially checked when the dialog box is created. This flag also indicates the state of the Read Only check box when the dialog box is closed.
cdlOFNShareAware	&H4000	Specifies that sharing violation errors will be ignored.

Color Dialog Box Flags

Constant	Value	Description
cdCCIFullOpen	&H2	Entire dialog box is displayed, including the Define Custom Colors section
cdlCCShowHelp	&H8	Causes the dialog box to display a Help button
cdlCCPreventFullOpen	&H4	Disables the Define Custom Colors command button and prevents the user from defining custom colors
cdlCCRGBInit	&H1	Sets the initial color value for the dialog box

Fonts Dialog Box Flags

Constant	Value	Description
cdlCFANSIOnly	&H400	Specifies that the dialog box allows only a selection of the fonts that use the Windows character set. If this flag is set, the user won't be able to select a font that contains only symbols.
cdlCFApply	&H200	Enables the Apply button on the dialog box.
cdlCFBoth	&H3	Causes the dialog box to list the available printer and screen fonts. The **hDC** property identifies the device context associated with the printer.
cdlCFEffects	&H100	Specifies that the dialog box enables strikethrough, underline, and color effects.
cdlCFFixedPitchOnly	&H4000	Specifies that the dialog box selects only fixed-pitch fonts.
cdlCFForceFontExist	&H10000	Specifies that an error message box is displayed if the user attempts to select a font or style that doesn't exist.
cdlCFHelpButton	&H4	Causes the dialog box to display a Help button.
cdlCFLimitSize	&H2000	Specifies that the dialog box selects only font sizes within the range specified by the **Min** and **Max** properties.

cdlCFNoFaceSel	&H80000	No font name selected.
cdlCFNoSimulations	&H1000	Specifies that the dialog box doesn't allow graphic device interface (GDI) font simulations.
cdlCFNoSizeSel	&H200000	No font size selected.
cdlCFNoStyleSel	&H100000	
cdlCFNoVectorFonts	&H800	Specifies that the dialog box doesn't allow vector-font selections.
cdlCFPrinterFonts	&H2	Causes the dialog box to list only the fonts supported by the printer, specified by the **hDC** property.
cdlCFScalableOnly	&H20000	Specifies that the dialog box allows only the selection of fonts that can be scaled.
cdlCFScreenFonts	&H1	Causes the dialog box to list only the screen fonts supported by the system.
cdlCFTTOnly	&H40000	Specifies that the dialog box allows only the selection of TrueType fonts.
cdlCFWYSIWYG	&H8000	Specifies that the dialog box allows only the selection of fonts that are available on both the printer and on screen. If this flag is set, the **cdlCFBoth** and **cdlCFScalableOnly** flags should also be set.

Printer Dialog Box Flags

Constant	Value	Description
cdlPDAllPages	&H0	Returns or sets the state of the All Pages option button.
cdlPDCollate	&H10	Returns or sets the state of the Collate check box.
cdlPDDisablePrintToFile	&H80000	Disables the Print To File check box.
cdlPDHelpButton	&H800	Causes the dialog box to display the Help button.
cdlPDHidePrintToFile	&H100000	Hides the Print To File check box.
cdlPDNoPageNums	&H8	Disables the Pages option button and the associated edit control.
cdlPDNoSelection	&H4	Disables the Selection option button.
cdlPDNoWarning	&H80	Prevents a warning message from being displayed when there is no default printer.
cdlPDPageNums	&H2	Returns or sets the state of the Pages option button.
cdlPDPrintSetup	&H40	Causes the system to display the Print Setup dialog box rather than the Print dialog box.
cdlPDPrintToFile	&H20	Returns or sets the state of the Print To File check box.
cdlPDReturnDC	&H100	Returns a device context for the printer selection made in the dialog box. The device context is returned in the dialog box's **hDC** property.
cdlPDReturnDefault	&H400	Returns default printer name.

cdlPDReturnIC	&H200	Returns an information context for the printer selection made in the dialog box. An information context provides a fast way to get information about the device without creating a device context. The information context is returned in the dialog box's **hDC** property.
cdlPDSelection	&H1	Returns or sets the state of the Selection option button. If neither **cdlPDPageNums** nor **cdlPDSelection** is specified, the All option button is in the selected state.
cdlPDUseDevModeCopies	&H40000	If a printer driver doesn't support multiple copies, setting this flag disables the copies edit control. If a driver does support multiple copies, setting this flag indicates that the dialog box stores the requested number of copies in the **Copies** property.

Help Constants

Constant	Value	Description
cdlHelpCommandHelp	&H102	Displays Help for a particular command
cdlHelpContents	&H3	Displays the contents topic in the current Help file
cdlHelpContext	&H1	Displays Help for a particular topic
cdlHelpContextPopup	&H8	Displays a topic identified by a context number
cdlHelpForceFile	&H9	Creates a Help file that displays text in only one font
cdlHelpHelpOnHelp	&H4	Displays Help for using the Help application itself
cdlHelpIndex	&H3	Displays the index of the specified Help file
cdlHelpKey	&H101	Displays Help for a particular keyword
cdlHelpPartialKey	&H105	Calls the search engine in Windows Help
cdlHelpQuit	&H2	Notifies the Help application that the specified Help file is no longer in use
cdlHelpSetContents	&H5	Designates a specific topic as the contents topic
cdlHelpSetIndex	&H5	Sets the current index for multi-index Help

Help Constants

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

Constant	Value	Description
cdlHelpContext	&H1	Displays Help for a particular topic
cdlHelpQuit	&H2	Notifies the Help application that the specified Help file is no longer in use
cdlHelpIndex	&H3	Displays the index of the specified Help file
cdlHelpContents	&H3	Displays the contents topic in the current Help file
cdlHelpHelpOnHelp	&H4	Displays Help for using the Help application itself
cdlHelpSetIndex	&H5	Sets the current index for multi-index Help
cdlHelpSetContents	&H5	Designates a specific topic as the contents topic
cdlHelpContextPopup	&H8	Displays a topic identified by a context number
cdlHelpForceFile	&H9	Creates a Help file that displays text in only one font
cdlHelpKey	&H101	Displays Help for a particular keyword
cdlHelpCommandHelp	&H102	Displays Help for a particular command
cdlHelpPartialKey	&H105	Calls the search engine in Windows Help

CommonDialog Error Constants

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

Constant	Value	Description
cdlAlloc	&H7FF0	Couldn't allocate memory for **FileName** or **Filter** property
cdlCancel	&H7FF3	Cancel was selected
cdlDialogFailure	&H8000	The function failed to load the dialog box
cdlFindResFailure	&H7FF9	The function failed to load a specified resource
cdlHelp	&H7FEF	Call to Windows Help failed
cdlInitialization	&H7FFD	The function failed during initialization
cdlLoadResFailure	&H7FF8	The function failed to load a specified string
cdlLockResFailure	&H7FF7	The function failed to lock a specified resource
cdlMemAllocFailure	&H7FF6	The function was unable to allocate memory for internal data structures
cdlMemLockFailure	&H7FF5	The function was unable to lock the memory associated with a handle
cdlNoFonts	&H5FFE	No fonts exist
cdlBufferTooSmall	&H4FFC	The buffer at which the member lpstrFile points is too small
cdlInvalidFileName	&H4FFD	Filename is invalid
cdlSubclassFailure	&H4FFE	An attempt to subclass a list box failed due to insufficient memory
cdlCreateICFailure	&H6FF5	The **PrintDlg** function failed when it attempted to create an information context
cdlDndmMismatch	&H6FF6	Data in the DevMode and DevNames data structures describe two different printers
cdlGetDevModeFail	&H6FFA	The printer device driver failed to initialize a DevMode data structure
cdlInitFailure	&H6FF9	The **PrintDlg** function failed during initialization
cdlLoadDrvFailure	&H6FFB	The **PrintDlg** function failed to load the specified printer's device driver
cdlLoadStrFailure	&H7FFA	The function failed to load a specified string.
cdlNoDefaultPrn	&H6FF7	A default printer doesn't exist
cdlNoDevices	&H6FF8	No printer device drivers were found
cdlParseFailure	&H6FFD	The **CommonDialog** function

		failed to parse the strings in the [devices] section of Win.ini
cdlPrinterCodes	&H6FFF	The PDReturnDefault flag was set, but either the hDevMode or hDevNames field was nonzero
cdlPrinterNotFound	&H6FF4	The [devices] section of Win.ini doesn't contain an entry for the requested printer
cdlRetDefFailure	&H6FFC	The PDReturnDefault flag was set, but either the hDevMode or hDevNames field was nonzero
cdlSetupFailure	&H6FFE	Failed to load required resources

Menu Control Constants

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

PopupMenu Method Alignment

Constant	Value	Description
vbPopupMenuLeftAlign	0	Pop-up menu left-aligned
vbPopupMenuCenterAlign	4	Pop-up menu centered
vbPopupMenuRightAlign	8	Pop-up menu right-aligned

PopupMenu Mouse Button Recognition

Constant	Value	Description
vbPopupMenuLeftButton	0	Pop-up menu recognizes left mouse button only
vbPopupMenuRightButton	2	Pop-up menu recognizes right and left mouse buttons

Menu Accelerator Constants

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

Constant	Value	Description
vbMenuAccelCtrlA	1	User-defined shortcut keystrokes
vbMenuAccelCtrlB	2	User-defined shortcut keystrokes
vbMenuAccelCtrlC	3	User-defined shortcut keystrokes
vbMenuAccelCtrlD	4	User-defined shortcut keystrokes
vbMenuAccelCtrlE	5	User-defined shortcut keystrokes
vbMenuAccelCtrlF	6	User-defined shortcut keystrokes
vbMenuAccelCtrlG	7	User-defined shortcut keystrokes
vbMenuAccelCtrlH	8	User-defined shortcut keystrokes
vbMenuAccelCtrlI	9	User-defined shortcut keystrokes
vbMenuAccelCtrlJ	10	User-defined shortcut keystrokes
vbMenuAccelCtrlK	11	User-defined shortcut keystrokes
vbMenuAccelCtrlL	12	User-defined shortcut keystrokes
vbMenuAccelCtrlM	13	User-defined shortcut keystrokes
vbMenuAccelCtrlN	14	User-defined shortcut keystrokes
vbMenuAccelCtrlO	15	User-defined shortcut keystrokes
vbMenuAccelCtrlP	16	User-defined shortcut keystrokes
vbMenuAccelCtrlQ	17	User-defined shortcut keystrokes
vbMenuAccelCtrlR	18	User-defined shortcut keystrokes
vbMenuAccelCtrlS	19	User-defined shortcut keystrokes
vbMenuAccelCtrlT	20	User-defined shortcut keystrokes
vbMenuAccelCtrlU	21	User-defined shortcut keystrokes
vbMenuAccelCtrlV	22	User-defined shortcut keystrokes
vbMenuAccelCtrlW	23	User-defined shortcut keystrokes
vbMenuAccelCtrlX	24	User-defined shortcut keystrokes
vbMenuAccelCtrlY	25	User-defined shortcut keystrokes
vbMenuAccelCtrlZ	26	User-defined shortcut keystrokes
vbMenuAccelF1	27	User-defined shortcut keystrokes
vbMenuAccelF2	28	User-defined shortcut keystrokes
vbMenuAccelF3	29	User-defined shortcut keystrokes
vbMenuAccelF4	30	User-defined shortcut keystrokes
vbMenuAccelF5	31	User-defined shortcut keystrokes
vbMenuAccelF6	32	User-defined shortcut keystrokes
vbMenuAccelF7	33	User-defined shortcut keystrokes
vbMenuAccelF8	34	User-defined shortcut keystrokes
vbMenuAccelF9	35	User-defined shortcut keystrokes
vbMenuAccelF11	36	User-defined shortcut keystrokes
vbMenuAccelF12	37	User-defined shortcut keystrokes
vbMenuAccelCtrlF1	38	User-defined shortcut keystrokes
vbMenuAccelCtrlF2	39	User-defined shortcut keystrokes
vbMenuAccelCtrlF3	40	User-defined shortcut keystrokes
vbMenuAccelCtrlF4	41	User-defined shortcut keystrokes

vbMenuAccelCtrlF5	42	User-defined shortcut keystrokes
vbMenuAccelCtrlF6	43	User-defined shortcut keystrokes
vbMenuAccelCtrlF7	44	User-defined shortcut keystrokes
vbMenuAccelCtrlF8	45	User-defined shortcut keystrokes
vbMenuAccelCtrlF9	46	User-defined shortcut keystrokes
vbMenuAccelCtrlF11	47	User-defined shortcut keystrokes
vbMenuAccelCtrlF12	48	User-defined shortcut keystrokes
vbMenuAccelShiftF1	49	User-defined shortcut keystrokes
vbMenuAccelShiftF2	50	User-defined shortcut keystrokes
vbMenuAccelShiftF3	51	User-defined shortcut keystrokes
vbMenuAccelShiftF4	52	User-defined shortcut keystrokes
vbMenuAccelShiftF5	53	User-defined shortcut keystrokes
vbMenuAccelShiftF6	54	User-defined shortcut keystrokes
vbMenuAccelShiftF7	55	User-defined shortcut keystrokes
vbMenuAccelShiftF8	56	User-defined shortcut keystrokes
vbMenuAccelShiftF9	57	User-defined shortcut keystrokes
vbMenuAccelShiftF11	58	User-defined shortcut keystrokes
vbMenuAccelShiftF12	59	User-defined shortcut keystrokes
vbMenuAccelShiftCtrlF1	60	User-defined shortcut keystrokes
vbMenuAccelShiftCtrlF2	61	User-defined shortcut keystrokes
vbMenuAccelShiftCtrlF3	62	User-defined shortcut keystrokes
vbMenuAccelShiftCtrlF4	63	User-defined shortcut keystrokes
vbMenuAccelShiftCtrlF5	64	User-defined shortcut keystrokes
vbMenuAccelShiftCtrlF6	65	User-defined shortcut keystrokes
vbMenuAccelShiftCtrlF7	66	User-defined shortcut keystrokes
vbMenuAccelShiftCtrlF8	67	User-defined shortcut keystrokes
vbMenuAccelShiftCtrlF9	68	User-defined shortcut keystrokes
vbMenuAccelShiftCtrlF11	69	User-defined shortcut keystrokes
vbMenuAccelShiftCtrlF12	70	User-defined shortcut keystrokes
vbMenuAccelCtrlIns	71	User-defined shortcut keystrokes
vbMenuAccelShiftIns	72	User-defined shortcut keystrokes
vbMenuAccelDel	73	User-defined shortcut keystrokes
vbMenuAccelShiftDel	74	User-defined shortcut keystrokes
vbMenuAccelAltBksp	75	User-defined shortcut keystrokes

Printer Object Constants

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

Printer Color Mode

Constant	Value	Description
vbPRCMMonochrome	1	Monochrome output
vbPRCMColor	2	Color output

Duplex Printing

Constant	Value	Description
vbPRDPSimplex	1	Single-sided printing
vbPRDPHorizontal	2	Double-sided horizontal printing
vbPRDPVertical	3	Double-sided vertical printing

Printer Orientation

Constant	Value	Description
vbPRORPortrait	1	Documents print with the top at the narrow side of the paper
vbPRORLandscape	2	Documents print with the top at the wide side of the paper

Print Quality

Constant	Value	Description
vbPRPQDraft	-1	Draft print quality
vbPRPQLow	-2	Low print quality
vbPRPQMedium	-3	Medium print quality
vbPRPQHigh	-4	High print quality

PaperBin Property

Constant	Value	Description
vbPRBNUpper	1	Use paper from the upper bin
vbPRBNLower	2	Use paper from the lower bin
vbPRBNMiddle	3	Use paper from the middle bin
vbPRBNManual	4	Wait for manual insertion of each sheet of paper
vbPRBNEnvelope	5	Use envelopes from the envelope feeder
vbPRBNEnvManual	6	Use envelopes from the envelope feeder, but wait for manual insertion
vbPRBNAuto	7	(Default) Use paper from the current default bin
vbPRBNTractor	8	Use paper fed from the tractor feeder
vbPRBNSmallFmt	9	Use paper from the small paper feeder
vbPRBNLargeFmt	10	Use paper from the large paper bin

vbPRBNLargeCapacity	11	Use paper from the large capacity feeder
vbPRBNCassette	14	Use paper from the attached cassette cartridge

PaperSize Property

Constant	Value	Description
vbPRPSLetter	1	Letter, 8 1/2 x 11 in
vbPRPSLetterSmall	2	+A611Letter Small, 8 1/2 x 11 in
vbPRPSTabloid	3	Tabloid, 11 x 17 in
vbPRPSLedger	4	Ledger, 17 x 11 in
vbPRPSLegal	5	Legal, 8 1/2 x 14 in
vbPRPSStatement	6	Statement, 5 1/2 x 8 1/2 in
vbPRPSExecutive	7	Executive, 7 1/2 x 10 1/2 in
vbPRPSA3	8	A3, 297 x 420 mm
vbPRPSA4	9	A4, 210 x 297 mm
vbPRPSA4Small	10	A4 Small, 210 x 297 mm
vbPRPSA5	11	A5, 148 x 210 mm
vbPRPSB4	12	B4, 250 x 354 mm
vbPRPSB5	13	B5, 182 x 257 mm
vbPRPSFolio	14	Folio, 8 1/2 x 13 in
vbPRPSQuarto	15	Quarto, 215 x 275 mm
vbPRPS1&H14	16	10 x 14 in
vbPRPS11x17	17	11 x 17 in
vbPRPSNote	18	Note, 8 1/2 x 11 in
vbPRPSEnv9	19	Envelope #9, 3 7/8 x 8 7/8 in
vbPRPSEnv10	20	Envelope #10, 4 1/8 x 9 1/2 in
vbPRPSEnv11	21	Envelope #11, 4 1/2 x 10 3/8 in
vbPRPSEnv12	22	Envelope #12, 4 1/2 x 11 in
vbPRPSEnv14	23	Envelope #14, 5 x 11 1/2 in
vbPRPSCSheet	24	C size sheet
vbPRPSDSheet	25	D size sheet
vbPRPSESheet	26	E size sheet
vbPRPSEnvDL	27	Envelope DL, 110 x 220 mm
vbPRPSEnvC3	29	Envelope C3, 324 x 458 mm
vbPRPSEnvC4	30	Envelope C4, 229 x 324 mm
vbPRPSEnvC5	28	Envelope C5, 162 x 229 mm
vbPRPSEnvC6	31	Envelope C6, 114 x 162 mm
vbPRPSEnvC65	32	Envelope C65, 114 x 229 mm
vbPRPSEnvB4	33	Envelope B4, 250 x 353 mm
vbPRPSEnvB5	34	Envelope B5, 176 x 250 mm
vbPRPSEnvB6	35	Envelope B6, 176 x 125 mm
vbPRPSEnvItaly	36	Envelope, 110 x 230 mm
vbPRPSEnvMonarch	37	Envelope Monarch, 3 7/8 x 7 1/2 in
vbPRPSEnvPersonal	38	Envelope, 3 5/8 x 6 1/2 in

vbPRPSFanfoldUS	39	U.S. Standard Fanfold, 14 7/8 x 11 in
vbPRPSFanfoldStdGerman	40	German Standard Fanfold, 8 1/2 x 12 in
vbPRPSFanfoldLglGerman	41	German Legal Fanfold, 8 1/2 x 13 in
vbPRPSUser	256	User-defined

RasterOp Constants

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

Constant	Value	Description
vbDstInvert	&H00550009	Inverts the destination bitmap
vbMergeCopy	&H00C000CA	Combines the pattern and the source bitmap
vbMergePaint	&H00BB0226	Combines the inverted source bitmap with the destination bitmap by using **Or**
vbNotSrcCopy	&H00330008	Copies the inverted source bitmap to the destination
vbNotSrcErase	&H001100A6	Inverts the result of combining the destination and source bitmaps by using **Or**
vbPatCopy	&H00F00021L	Copies the pattern to the destination bitmap
vbPatInvert	&H005A0049L	Combines the destination bitmap with the pattern by using **Xor**
vbPatPaint	&H00FB0A09L	Combines the inverted source bitmap with the pattern by using **Or**. Combines the result of this operation with the destination bitmap by using **Or**
vbSrcAnd	&H008800C6	Combines pixels of the destination and source bitmaps by using **And**
vbSrcCopy	&H00CC0020	Copies the source bitmap to the destination bitmap
vbSrcErase	&H00440328	Inverts the destination bitmap and combines the result with the source bitmap by using **And**
vbSrcInvert	&H00660046	Combines pixels of the destination and source bitmaps by using **Xor**
vbSrcPaint	&H00EE0086	Combines pixels of the destination and source bitmaps by using **Or**

VBTranslateColor/OLETranslateColor Constants

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

Constant	Value	Description
vbInactiveCaptionText	&H80000013	Color of text in an inactive caption
vb3DHighlight	&H80000014	Highlight color for 3-D display elements
vb3DFace	&H8000000F	Dark shadow color for 3-D display elements
vbMsgBox	&H80000017	Background color for message boxes and system dialog boxes
vbMsgBoxText	&H80000018	Color of text displayed in message boxes and system dialog boxes
vb3DShadow	&H80000010	Color of automatic window shadows
vb3DDKShadow	&H80000015	Darkest shadow
vb3DLight	&H80000016	Second lightest of the 3-D colors (after **vb3DHighlight**)

Activate, Deactivate Events

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtActivateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtActivateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtActivateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtActivateS"}

- Activate — occurs when an object becomes the active window.
- Deactivate — occurs when an objectis no longer the active window.

Syntax

Private Sub *object*_**Activate( )**
Private Sub *object*_**Deactivate( )**

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

Remarks

An object can become active by user action, such as clicking it, or by using the **Show** or **SetFocus** methods in code.

The Activate event can occur only when an object is visible. For example, a form loaded with the **Load** statement isn't visible unless you use the **Show** method or set the form's **Visible** property to **True**.

The Activate and Deactivate events occur only when moving the focus within an application. Moving the focus to or from an object in another application doesn't trigger either event. The Deactivate event doesn't occur when unloading an object.

The Activate event occurs before the GotFocus event; the LostFocus event occurs before the Deactivate event.

These events occur for MDI child forms only when the focus changes from one child form to another. In an **MDIForm** object with two child forms, for example, the child forms receive these events when the focus moves between them. However, when the focus changes between a child form and a non-MDI child form, the parent **MDIForm** receives the Activate and Deactivate events.

If an .exe file built by Visual Basic displays a dialog box created by a .dll file also built in Visual Basic, the .exe file's form will get the Deactivate and LostFocus events. This may be unexpected, because you should not get the Deactivate event:

- If the object is an out-of-process component.
- If the object isn't written in Visual Basic.
- In the development environment when calling a DLL built in Visual Basic.

Change Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtChangeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtChangeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtChangeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtChangeS"}

Indicates the contents of a control have changed. How and when this event occurs varies with the control:

- **ComboBox** — changes the text in the text box portion of the control. Occurs only if the **Style** property is set to 0 (Dropdown Combo) or 1 (Simple Combo) and the user changes the text or you change the **Text** property setting through code.
- **DirListBox** — changes the selected directory. Occurs when the user double-clicks a new directory or when you change the **Path** property setting through code.
- **DriveListBox** — changes the selected drive. Occurs when the user selects a new drive or when you change the **Drive** property setting through code.
- **HScrollBar** and **VScrollBar** (horizontal and vertical scroll bars) — move the scroll box portion of the scroll bar. Occurs when the user scrolls or when you change the **Value** property setting through code.
- **Label** — changes the contents of the **Label**. Occurs when a DDE link updates data or when you change the **Caption** property setting through code.
- **PictureBox** — changes the contents of the **PictureBox**. Occurs when a DDE link updates data or when you change the **Picture** property setting through code.
- **TextBox** — changes the contents of the text box. Occurs when a DDE link updates data, when a user changes the text, or when you change the **Text** property setting through code.

Syntax

Private Sub *object*_**Change(**[*index* **As Integer**]**)**

The Change event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array

Remarks

The Change event procedure can synchronize or coordinate data display among controls. For example, you can use a scroll bar's Change event procedure to update the scroll bar's **Value** property setting in a **TextBox** control. Or you can use a Change event procedure to display data and formulas in a work area and results in another area.

Change event procedures are also useful for updating properties in file-system controls (**DirListBox**, **DriveListBox**, and **FileListBox**). For example, you can update the **Path** property setting for a **DirListBox** control to reflect a change in a **DriveListBox** control's **Drive** property setting.

Note A Change event procedure can sometimes cause a cascading event. This occurs when the control's Change event alters the control's contents, for example, by setting a property in code that determines the control's value, such as the **Text** property setting for a **TextBox** control. To prevent a cascading event:

- If possible, avoid writing a Change event procedure for a control that alters that control's contents. If you do write such a procedure, be sure to set a flag that prevents further changes while the current change is in progress.
- Avoid creating two or more controls whose Change event procedures affect each other, for

example, two **TextBox** controls that update each other during their Change events.

- Avoid using a **MsgBox** function or statement in this event for **HScrollBar** and **VScrollBar** controls.

Click Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtClickC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtClickX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtClickA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtClickS"}

Occurs when the user presses and then releases a mouse button over an object. It can also occur when the value of a control is changed.

For a **Form** object, this event occurs when the user clicks either a blank area or a disabled control. For a control, this event occurs when the user:

- Clicks a control with the left or right mouse button. With a **CheckBox**, **CommandButton**, **Listbox**, or **OptionButton** control, the Click event occurs only when the user clicks the left mouse button.
- Selects an item in a **ComboBox** or **ListBox** control, either by pressing the arrow keys or by clicking the mouse button.
- Presses the SPACEBAR when a **CommandButton**, **OptionButton**, or **CheckBox** control has the focus.
- Presses ENTER when a form has a **CommandButton** control with its **Default** property set to **True**.
- Presses ESC when a form has a Cancel button — a **CommandButton** control with its **Cancel** property set to **True**.
- Presses an access key for a control. For example, if the caption of a **CommandButton** control is "&Go", pressing ALT+G triggers the event.

You can also trigger the Click event in code by:

- Setting a **CommandButton** control's **Value** property to **True**.
- Setting an **OptionButton** control's **Value** property to **True**.
- Changing a **CheckBox** control's **Value** property setting.

Syntax

Private Sub Form_Click( )
Private Sub *object*_**Click(**[*index* **As Integer**]**)**

The Click event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.

Remarks

Typically, you attach a Click event procedure to a **CommandButton** control, **Menu** object, or **PictureBox** control to carry out commands and command-like actions. For the other applicable controls, use this event to trigger actions in response to a change in the control.

You can use a control's **Value** property to test the state of the control from code. Clicking a control generates MouseDown and MouseUp events in addition to the Click event. The order in which these three events occur varies from control to control. For example, for **ListBox** and **CommandButton** controls, the events occur in this order: MouseDown, Click, MouseUp. But for **FileListBox**, **Label**, or **PictureBox** controls, the events occur in this order: MouseDown, MouseUp, and Click. When you're attaching event procedures for these related events, be sure that their actions don't conflict. If the order of events is important in your application, test the control to determine the event order.

Note To distinguish between the left, right, and middle mouse buttons, use the MouseDown and MouseUp events.

If there is code in the Click event, the DlbClick event will never trigger, because the Click event is the first event to trigger between the two. As a result, the mouse click is intercepted by the Click event, so the DblClick event doesn't occur.

DragDrop Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtDragDropC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtDragDropX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtDragDropA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtDragDropS"}

Occurs when a drag-and-drop operation is completed as a result of dragging a control over an object and releasing the mouse button or using the **Drag** method with its *action* argument set to 2 (Drop).

Syntax

Private Sub Form_DragDrop(*source* **As Control**, *x* **As Single**, *y* **As Single)**
Private Sub MDIForm_DragDrop(*source* **As Control**, *x* **As Single**, *y* **As Single)**
Private Sub *object***_DragDrop(**[*index* **As Integer**,]*source* **As Control**, *x* **As Single**, *y* **As Single)**

The DragDrop event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.
source	The control being dragged. You can include properties and methods in the event procedure with this argument — for example, `Source.Visible = 0`.
x, *y*	A number that specifies the current horizontal (*x*) and vertical (*y*) position of the mouse pointer within the target form or control. These coordinates are always expressed in terms of the target's coordinate system as set by the **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, and **ScaleTop** properties.

Remarks

Use a DragDrop event procedure to control what happens after a drag operation is completed. For example, you can move the source control to a new location or copy a file from one location to another.

When multiple controls can potentially be used in a *source* argument:

- Use the **TypeOf** keyword with the **If** statement to determine the type of control used with *source.*
- Use the control's **Tag** property to identify a control, and then use a DragDrop event procedure.

Note Use the **DragMode** property and **Drag** method to specify the way dragging is initiated. Once dragging has been initiated, you can handle events that precede a DragDrop event with a DragOver event procedure.

DragOver Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtDragOverC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtDragOverX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtDragOverA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtDragOverS"}

Occurs when a drag-and-drop operation is in progress. You can use this event to monitor the mouse pointer as it enters, leaves, or rests directly over a valid target. The mouse pointer position determines the target object that receives this event.

Syntax

Private Sub Form_DragOver(*source* **As Control**, *x* **As Single**, *y* **As Single**, *state* **As Integer)**
Private Sub MDIForm_DragOver(*source* **As Control**, *x* **As Single**, *y* **As Single**, *state* **As Integer)**
Private Sub *object*_**DragOver(**[*index* **As Integer**,]*source* **As Control**, *x* **As Single**, *y* **As Single**, *state* **As Integer)**

The DragOver event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.
source	The control being dragged. You can refer to properties and methods in the event procedure with this argument — for example, `Source.Visible = False`.
x, *y*	A number that specifies the current horizontal (*x*) and vertical (*y*) position of the mouse pointer within the target form or control. These coordinates are always expressed in terms of the target's coordinate system as set by the **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, and **ScaleTop** properties.
state	An integer that corresponds to the transition state of the control being dragged in relation to a target form or control: 0 = Enter (source control is being dragged within the range of a target). 1 = Leave (source control is being dragged out of the range of a target). 2 = Over (source control has moved from one position in the target to another).

Remarks

Use a DragOver event procedure to determine what happens after dragging is initiated and before a control drops onto a target. For example, you can verify a valid target range by highlighting the target (set the **BackColor** or **ForeColor** property from code) or by displaying a special drag pointer (set the **DragIcon** or **MousePointer** property from code).

Use the *state* argument to determine actions at key transition points. For example, you might highlight a possible target when *state* is set to 0 (Enter) and restore the object's previous appearance when *state* is set to 1 (Leave).

When an object receives a DragOver event while *state* is set to 0 (Enter):

- If the source control is dropped on the object, that object receives a DragDrop event.
- If the source control isn't dropped on the object, that object receives another DragOver event when *state* is set to 1 (Leave).

Note Use the **DragMode** property and **Drag** method to specify the way dragging is initiated. For suggested techniques with the *source* argument, see Remarks for the DragDrop event topic.

DropDown Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtDropDownC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtDropDownX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtDropDownA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtDropDownS"}

Occurs when the list portion of a **ComboBox** control is about to drop down; this event doesn't occur if a **ComboBox** control's **Style** property is set to 1 (Simple Combo).

Syntax

Private Sub *object*_**DropDown(**[*index* **As Integer**]**)**

The DropDown event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.

Remarks

Use a DropDown event procedure to make final updates to a **ComboBox** list before the user makes a selection. This enables you to add or remove items from the list using the **AddItem** or **RemoveItem** methods. This flexibility is useful when you want some interplay between controls — for example, if what you want to load into a **ComboBox** list depends on what the user selects in an **OptionButton** group.

GotFocus Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtGotFocusC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtGotFocusX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtGotFocusA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtGotFocusS"}

Occurs when an object receives the focus, either by user action, such as tabbing to or clicking the object, or by changing the focus in code using the **SetFocus** method. A form receives the focus only when all visible controls are disabled.

Syntax

Private Sub Form_GotFocus( )
Private Sub *object*_**GotFocus(**[*index* **As Integer**]**)**

The GotFocus event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.

Remarks

Typically, you use a GotFocus event procedure to specify the actions that occur when a control or form first receives the focus. For example, by attaching a GotFocus event procedure to each control on a form, you can guide the user by displaying brief instructions or status bar messages. You can also provide visual cues by enabling, disabling, or showing other controls that depend on the control that has the focus.

Note An object can receive the focus only if its **Enabled** and **Visible** properties are set to **True**. To customize the keyboard interface in Visual Basic for moving the focus, set the tab order or specify access keys for controls on a form.

KeyDown, KeyUp Events

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtKeyDownC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtKeyDownX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtKeyDownA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtKeyDownS"}

Occur when the user presses (KeyDown) or releases (KeyUp) a key while an object has the focus. (To interpret ANSI characters, use the KeyPress event.)

Syntax

Private Sub Form_KeyDown(*keycode* **As Integer**, *shift* **As Integer)**
Private Sub *object*_**KeyDown(**[*index* **As Integer**,]*keycode* **As Integer**, *shift* **As Integer)**
Private Sub Form_KeyUp(*keycode* **As Integer**, *shift* **As Integer)**
Private Sub *object*_**KeyUp(**[*index* **As Integer**,]*keycode* **As Integer**, *shift* **As Integer)**

The KeyDown and KeyUp event syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.
keycode	A key code, such as **vbKeyF1** (the F1 key) or **vbKeyHome** (the HOME key). To specify key codes, use the constants in the Visual Basic (VB) object library in the Object Browser.
shift	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The *shift* argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of *shift* is 6.

Remarks

For both events, the object with the focus receives all keystrokes. A form can have the focus only if it has no visible and enabled controls. Although the KeyDown and KeyUp events can apply to most keys, they're most often used for:

- Extended character keys such as function keys.
- Navigation keys.
- Combinations of keys with standard keyboard modifiers.
- Distinguishing between the numeric keypad and regular number keys.

Use KeyDown and KeyUp event procedures if you need to respond to both the pressing and releasing of a key.

KeyDown and KeyUp aren't invoked for:

- The ENTER key if the form has a **CommandButton** control with the **Default** property set to **True**.
- The ESC key if the form has a **CommandButton** control with the **Cancel** property set to **True**.
- The TAB key.

KeyDown and KeyUp interpret the uppercase and lowercase of each character by means of two arguments: *keycode*, which indicates the physical key (thus returning A and a as the same key) and *shift*, which indicates the state of *shift*+*key* and therefore returns either A or a.

If you need to test for the *shift* argument, you can use the *shift* constants which define the bits within the argument. The constants have the following values:

Constant	Value	Description
vbShiftMask	1	SHIFT key bit mask.
VbCtrlMask	2	CTRL key bit mask.
VbAltMask	4	ALT key bit mask.

The constants act as bit masks that you can use to test for any combination of keys.

You test for a condition by first assigning each result to a temporary integer variable and then comparing *shift* to a bit mask. Use the **And** operator with the *shift* argument to test whether the condition is greater than 0, indicating that the modifier was pressed, as in this example:

```
ShiftDown = (Shift And vbShiftMask) > 0
```

In a procedure, you can test for any combination of conditions, as in this example:

```
If ShiftDown And CtrlDown Then
```

Note If the **KeyPreview** property is set to **True**, a form receives these events before controls on the form receive the events. Use the **KeyPreview** property to create global keyboard-handling routines.

KeyPress Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtKeyPressC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtKeyPressX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtKeyPressA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtKeyPressS"}

Occurs when the user presses and releases an ANSI key.

Syntax

Private Sub Form_KeyPress(*keyascii* **As Integer)**
Private Sub *object*_**KeyPress(**[*index* **As Integer**,]*keyascii* **As Integer)**

The KeyPress event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.
keyascii	An integer that returns a standard numeric ANSI keycode. *Keyascii* is passed by reference; changing it sends a different character to the object. Changing *keyascii* to 0 cancels the keystroke so the object receives no character.

Remarks

The object with the focus receives the event. A form can receive the event only if it has no visible and enabled controls or if the **KeyPreview** property is set to **True**. A KeyPress event can involve any printable keyboard character, the CTRL key combined with a character from the standard alphabet or one of a few special characters, and the ENTER or BACKSPACE key. A KeyPress event procedure is useful for intercepting keystrokes entered in a **TextBox** or **ComboBox** control. It enables you to immediately test keystrokes for validity or to format characters as they're typed. Changing the value of the *keyascii* argument changes the character displayed.

You can convert the *keyascii* argument into a character by using the expression:

```
Chr(KeyAscii)
```

You can then perform string operations and translate the character back to an ANSI number that the control can interpret by using the expression:

```
KeyAscii = Asc(char)
```

Use KeyDown and KeyUp event procedures to handle any keystroke not recognized by KeyPress, such as function keys, editing keys, navigation keys, and any combinations of these with keyboard modifiers. Unlike the KeyDown and KeyUp events, KeyPress doesn't indicate the physical state of the keyboard; instead, it passes a character.

KeyPress interprets the uppercase and lowercase of each character as separate key codes and, therefore, as two separate characters. KeyDown and KeyUp interpret the uppercase and lowercase of each character by means of two arguments: *keycode*, which indicates the physical key (thus returning A and a as the same key), and *shift*, which indicates the state of *shift+key* and therefore returns either A or a.

If the **KeyPreview** property is set to **True**, a form receives the event before controls on the form receive the event. Use the **KeyPreview** property to create global keyboard-handling routines.

Note The ANSI number for the keyboard combination of CTRL+@ is 0. Because Visual Basic recognizes a *keyascii* value of 0 as a zero-length string (""), avoid using CTRL+@ in your applications.

LinkClose Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtLinkCloseC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtLinkCloseX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtLinkCloseA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtLinkCloseS"}

Occurs when a DDE conversation terminates. Either application in a DDE conversation may terminate a conversation at any time.

Syntax

Private Sub Form_LinkClose( )
Private Sub MDIForm_LinkClose( )
Private Sub *object*_**LinkClose(**[*index* **As Integer**]**)**

The LinkClose event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.

Remarks

Typically, you use a LinkClose event procedure to notify the user that a DDE conversation has been terminated. You can also include troubleshooting information on reestablishing a connection or where to go for assistance. For brief messages, use the **MsgBox** function.

LinkError Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtLinkErrorC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtLinkErrorX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtLinkErrorA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtLinkErrorS"}

Occurs when there is an error during a DDE conversation. This event is recognized only as the result of a DDE-related error that occurs when no Visual Basic code is being executed. The error number is passed as an argument.

Syntax

Private Sub Form_LinkError(*linkerr* **As Integer)**
Private Sub MDIForm_LinkError(*linkerr* **As Integer)**
Private Sub *object*_**LinkError(**[*index* **As Integer**,]*linkerr* **As Integer)**

The LinkError event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
linkerr	Error number of the DDE-related error, as described in Return Values.
index	An integer that uniquely identifies a control if it's in a control array.

Return Values

The following table lists all error numbers returned for the *linkerr* argument and a brief explanation of each error:

Value	Description
1	The other application has requested data in the wrong format. This error may occur several times in succession as Visual Basic tries to find a format the other application recognizes.
6	The destination application attempted to continue a DDE conversation after you set the **LinkMode** property on your source form to 0 (None).
7	All the source links are in use (there is a limit of 128 links per source).
8	For destination controls: An automatic link or **LinkRequest** method failed to update the data in the control. For source forms: The destination attempted to poke data to a control and the attempt failed.
11	Not enough memory for DDE.

Remarks

Use a LinkError event procedure to notify the user of the particular error that has occurred. You can also include code to fix the problem or troubleshooting information on reestablishing a connection or on where to go for assistance. For brief messages, use the **MsgBox** function.

LinkExecute Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtLinkExecuteC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtLinkExecuteX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtLinkExecuteA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtLinkExecuteS"}

Occurs when a command string is sent by a destination application in a DDE conversation. The destination application expects the source application to perform the operation described by the string.

Syntax

Private Sub *object*_**LinkExecute(***cmdstr* **As String**, *cancel* **As Integer)**

The LinkExecute event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
cmdstr	The command string expression sent by the destination application.
cancel	An integer that tells the destination whether the command string was accepted or refused. Setting *cancel* to 0 informs the destination that the command string was accepted. Setting *cancel* to any nonzero value informs the destination that the command string was rejected. (The default is set to -1, indicating *cancel*.)

Remarks

There is no required syntax for *cmdstr*. How your application responds to different strings is completely up to you.

If you haven't created a LinkExecute event procedure, Visual Basic rejects command strings from destination applications.

LinkNotify Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtLinkNotifyC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtLinkNotifyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtLinkNotifyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtLinkNotifyS"}

Occurs when the source has changed the data defined by the DDE link if the **LinkMode** property of the destination control is set to 3 (Notify).

Syntax

Private Sub *object*_**LinkNotify(**[*index* **As Integer**]**)**

The LinkNotify event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.

Remarks

Typically, in the LinkNotify event your code notifies the user, gets the new data immediately, or defers getting the data until later. You can use the **LinkRequest** method to obtain the new data from the source.

LinkOpen Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtLinkOpenC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtLinkOpenX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtLinkOpenA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtLinkOpenS"}

Occurs when a DDE conversation is being initiated.

Syntax

Private Sub Form_LinkOpen(*cancel* **As Integer)**
Private Sub MDIForm_LinkOpen(*cancel* **As Integer)**
Private Sub *object*__**LinkOpen(**[*index* **As Integer**,]*cancel* **As Integer)**

The LinkOpen event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
cancel	An integer that determines whether the DDE conversation is established or not. Leaving *cancel* set to 0 (the default) establishes the conversation. Setting *cancel* to any nonzero value refuses the conversation.
index	An integer that uniquely identifies a control if it's in a control array.

Remarks

This event occurs for forms when a destination application is initiating a DDE conversation with the form. It occurs for controls when a control is initiating a DDE conversation with a source application.

Load Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtLoadC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtLoadX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtLoadA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtLoadS"}

Occurs when a form is loaded. For a startup form, occurs when an application starts as the result of a **Load** statement or as the result of a reference to an unloaded form's properties or controls.

Syntax

Private Sub Form_Load( )
Private Sub MDIForm_Load( )

Remarks

Typically, you use a Load event procedure to include initialization code for a form — for example, code that specifies default settings for controls, indicates contents to be loaded into **ComboBox** or **ListBox** controls, and initializes form-level variables.

The Load event occurs after the Initialize event.

When you reference a property of an unloaded form in code, the form is automatically loaded but isn't automatically made visible unless the **MDIChild** property is set to **True**. If an **MDIForm** object isn't loaded and an MDI child form is loaded, both the **MDIForm** and the child form are automatically loaded and both become visible. Other forms aren't shown until you either use the **Show** method or set the **Visible** property to **True**.

The following code in an **MDIForm** Load event automatically loads an MDI child form (assuming `Form1` has its **MDIChild** property set to **True**):

```
Dim NewForm As New Form1
NewForm.Caption = "New Form" ' Loads form by reference.
```

Because all child forms become visible when loaded, the reference to the **Caption** property loads the form and makes it visible.

Note When you create procedures for related events, such as Activate, GotFocus, Paint, and Resize, be sure that their actions don't conflict and that they don't cause recursive events.

LostFocus Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtLostFocusC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtLostFocusX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtLostFocusA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtLostFocusS"}

Occurs when an object loses the focus, either by user action, such as tabbing to or clicking another object, or by changing the focus in code using the **SetFocus** method.

Syntax

Private Sub Form_LostFocus( )
Private Sub *object*_**LostFocus(**[*index* **As Integer**]**)**

The LostFocus event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.

Remarks

A LostFocus event procedure is primarily useful for verification and validation updates. Using LostFocus can cause validation to take place as the user moves the focus from the control. Another use for this type of event procedure is enabling, disabling, hiding, and displaying other objects as in a GotFocus event procedure. You can also reverse or change conditions that you set up in the object's GotFocus event procedure.

If an .exe file built by Visual Basic displays a dialog box created by a .dll file also built in Visual Basic, the .exe file's form will get Deactivate and LostFocus events. This may be unexpected, because you should not get the Deactivate event:

- If the object is an out-of-process component.
- If the object isn't written in Visual Basic.
- In the development environment when calling a DLL built in Visual Basic.

MouseDown, MouseUp Events

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtMouseDownC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtMouseDownX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtMouseDownA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtMouseDownS"}

Occur when the user presses (MouseDown) or releases (MouseUp) a mouse button.

Syntax

Private Sub Form_MouseDown(*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**
Private Sub MDIForm_MouseDown(*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**
Private Sub *object***_MouseDown([***index* **As Integer**,]*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**
Private Sub Form_MouseUp(*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**
Private Sub MDIForm_MouseUp(*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**
Private Sub *object* **_MouseUp([***index* **As Integer**,]*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**

The MouseDown and MouseUp event syntaxes have these parts:

Part	Description
object	Returns an object expression that evaluates to an object in the Applies To list.
index	Returns an integer that uniquely identifies a control if it's in a control array.
button	Returns an integer that identifies the button that was pressed (MouseDown) or released (MouseUp) to cause the event. The *button* argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Only one of the bits is set, indicating the button that caused the event.
shift	Returns an integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the *button* argument is pressed or released. A bit is set if the key is down. The *shift* argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The *shift* argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of *shift* would be 6.
x, *y*	Returns a number that specifies the current location of the mouse pointer. The *x* and *y* values are always expressed in terms of the coordinate system set by the **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, and **ScaleTop** properties of the object.

Remarks

Use a MouseDown or MouseUp event procedure to specify actions that will occur when a given mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events enable you to distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers.

The following applies to both Click and DblClick events:

- If a mouse button is pressed while the pointer is over a form or control, that object "captures" the mouse and receives all mouse events up to and including the last MouseUp event. This implies that the *x*, *y* mouse-pointer coordinates returned by a mouse event may not always be in the internal area of the object that receives them.
- If mouse buttons are pressed in succession, the object that captures the mouse after the first press receives all mouse events until all buttons are released.

If you need to test for the *button* or *shift* arguments, you can use constants listed in the Visual Basic (VB) object library in the Object Browser to define the bits within the argument:

Constant (Button)	Value	Description
vbLeftButton	1	Left button is pressed
vbRightButton	2	Right button is pressed
vbMiddleButton	4	Middle button is pressed

Constant (Shift)	Value	Description
vbShiftMask	1	SHIFT key is pressed.
vbCtrlMask	2	CTRL key is pressed.
vbAltMask	4	ALT key is pressed.

The constants then act as bit masks you can use to test for any combination of buttons without having to figure out the unique bit field value for each combination.

Note You can use a MouseMove event procedure to respond to an event caused by moving the mouse. The *button* argument for MouseDown and MouseUp differs from the *button* argument used for MouseMove. For MouseDown and MouseUp, the *button* argument indicates exactly one button per event, whereas for MouseMove, it indicates the current state of all buttons.

MouseMove Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtMouseMoveC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtMouseMoveX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtMouseMoveA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtMouseMoveS"}

Occurs when the user moves the mouse.

Syntax

Private Sub Form_MouseMove(*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**
Private Sub MDIForm_MouseMove(*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**
Private Sub *object*_**MouseMove([***index* **As Integer**,] *button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**

The MouseMove event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.
button	An integer that corresponds to the state of the mouse buttons in which a bit is set if the button is down. The *button* argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. It indicates the complete state of the mouse buttons; some, all, or none of these three bits can be set, indicating that some, all, or none of the buttons are pressed.
shift	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys. A bit is set if the key is down. The *shift* argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The *shift* argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of *shift* would be 6.
x, *y*	A number that specifies the current location of the mouse pointer. The *x* and *y* values are always expressed in terms of the coordinate system set by the **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, and **ScaleTop** properties of the object.

Remarks

The MouseMove event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseMove event whenever the mouse position is within its borders.

If you need to test for the *button* or *shift* arguments, you can use constants listed in the Visual Basic (VB) object library in the Object Browser to define the bits within the argument:

Constant (Button)	Value	Description
vbLeftButton	1	Left button is pressed.
vbRightButton	2	Right button is pressed.
vbMiddleButton	4	Middle button is pressed.

Constant (Shift)	Value	Description
vbShiftMask	1	SHIFT key is pressed.
vbCtrlMask	2	CTRL key is pressed.
vbAltMask	4	ALT key is pressed.

The constants then act as bit masks you can use to test for any combination of buttons without having to figure out the unique bit field value for each combination.

You test for a condition by first assigning each result to a temporary integer variable and then comparing the *button* or *shift* arguments to a bit mask. Use the **And** operator with each argument to test if the condition is greater than zero, indicating the key or button is pressed, as in this example:

```
LeftDown = (Button And vbLeftButton) > 0
CtrlDown = (Shift And vbCtrlMask) > 0
```

Then, in a procedure, you can test for any combination of conditions, as in this example:

```
If LeftDown And CtrlDown Then
```

Note You can use MouseDown and MouseUp event procedures to respond to events caused by pressing and releasing mouse buttons.

The *button* argument for MouseMove differs from the *button* argument for MouseDown and MouseUp. For MouseMove, the *button* argument indicates the current state of all buttons; a single MouseMove event can indicate that some, all, or no buttons are pressed. For MouseDown and MouseUp, the *button* argument indicates exactly one button per event.

Any time you move a window inside a MouseMove event, it can cause a cascading event. MouseMove events are generated when the window moves underneath the pointer. A MouseMove event can be generated even if the mouse is perfectly stationary.

Paint Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtPaintC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtPaintX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtPaintA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtPaintS"}

Occurs when part or all of an object is exposed after being moved or enlarged, or after a window that was covering the object has been moved.

Syntax

Private Sub Form_Paint( )
Private Sub *object*_**Paint(**[*index* **As Integer**]**)**

The Paint event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.

Remarks

A Paint event procedure is useful if you have output from graphics methods in your code. With a Paint procedure, you can ensure that such output is repainted when necessary.

The Paint event is invoked when the **Refresh** method is used. If the **AutoRedraw** property is set to **True**, repainting or redrawing is automatic, so no Paint events are necessary.

If the **ClipControls** property is set to **False**, graphics methods in the Paint event procedure affect only newly exposed areas of the form; otherwise, the graphics methods repaint all areas of the form not covered by controls (except **Image**, **Label**, **Line**, and **Shape** controls).

Using a **Refresh** method in a Resize event procedure forces repainting of the entire object every time a user resizes the form.

Note Using a Paint event procedure for certain tasks can cause a cascading event. In general, avoid using a Paint event procedure to do the following:

- Move or size a form or control.
- Change any variables that affect size or appearance, such as setting an object's **BackColor** property.
- Invoke a **Refresh** method.

A Resize event procedure may be more appropriate for some of these tasks.

PathChange Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtPathChangeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtPathChangeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtPathChangeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtPathChangeS"}

Occurs when the path is changed by setting the **FileName** or **Path** property in code.

Syntax

Private Sub *object*_**PathChange(**[*index* **As Integer**]**)**

The PathChange event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.

Remarks

You can use a PathChange event procedure to respond to path changes in a **FileListBox** control. When you assign a string containing a new path to the **FileName** property, the **FileListBox** control invokes the PathChange event.

PatternChange Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtPatternChangeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtPatternChangeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtPatternChangeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtPatternChangeS"}

Occurs when the file listing pattern, such as **"*.*"**, is changed by setting the **FileName** or **Pattern** property in code.

Syntax

Private Sub *object*_**PatternChange(**[*index* **As Integer**]**)**

The PatternChange event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.

Remarks

You can use a PatternChange event procedure to respond to pattern changes in a **FileListBox** control. When you assign a string containing a new pattern to the **FileName** property, the **FileListBox** invokes the PathChange event.

QueryUnload Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtQueryUnloadC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtQueryUnloadX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtQueryUnloadA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtQueryUnloadS"}

Occurs before a form or application closes. When an **MDIForm** object closes, the QueryUnload event occurs first for the MDI form and then in all MDI child forms. If no form cancels the QueryUnload event, the Unload event occurs first in all other forms and then in an MDI form. When a child form or a **Form** object closes, the QueryUnload event in that form occurs before the form's Unload event.

Syntax

Private Sub Form_QueryUnload(*cancel* **As Integer**, *unloadmode* **As Integer)**
Private Sub MDIForm_QueryUnload(*cancel* **As Integer**, *unloadmode* **As Integer)**

The QueryUnload event syntax has these parts:

Part	Description
cancel	An integer. Setting this argument to any value other than 0 stops the QueryUnload event in all loaded forms and stops the form and application from closing.
unloadmode	A value or constant indicating the cause of the QueryUnload event, as described in Return Values.

Return Values

The *unloadmode* argument returns the following values:

Constant	Value	Description
vbFormControlMenu	0	The user chose the Close command from the Control menu on the form.
vbFormCode	1	The **Unload** statement is invoked from code.
vbAppWindows	2	The current Microsoft Windows operating environment session is ending.
vbAppTaskManager	3	The Microsoft Windows Task Manager is closing the application.
vbFormMDIForm	4	An MDI child form is closing because the MDI form is closing.

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

Remarks

This event is typically used to make sure there are no unfinished tasks in the forms included in an application before that application closes. For example, if a user has not yet saved some new data in any form, your application can prompt the user to save the data.

When an application closes, you can use either the QueryUnload or Unload event procedure to set the **Cancel** property to **True**, stopping the closing process. However, the QueryUnload event occurs in all forms before any are unloaded, and the Unload event occurs as each form is unloaded.

Resize Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtResizeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtResizeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtResizeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtResizeS"}

Occurs when an object is first displayed or when the window state of an object changes. (For example, a form is maximized, minimized, or restored.)

Syntax

Private Sub Form_Resize( )
Private Sub *object*_**Resize(***height* **As Single**, *width* **As Single)**

The Resize event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
height	Number specifying the new height of the control.
width	Number specifying the new width of the control.

Remarks

Use a Resize event procedure to move or resize controls when the parent form is resized. You can also use this event procedure to recalculate variables or properties, such as **ScaleHeight** and **ScaleWidth**, that may depend on the size of the form. If you want graphics to maintain sizes proportional to the form when it's resized, invoke the Paint event by using the **Refresh** method in a Resize event procedure.

Whenever the **AutoRedraw** property is set to **False** and the form is resized, Visual Basic also calls the related events, Resize and Paint, in that order. When you attach procedures for these related events, be sure their actions don't conflict.

When an **OLE** container control's **SizeMode** property is set to 2 (Autosize), the control is automatically sized according to the display size of the object contained in the control. If the display size of the object changes, the control is automatically resized to fit the object. When this occurs, the Resize event is invoked for the object before the **OLE** container control is resized. The *height* and *width* parts indicate the optimal size for displaying the object (this size is determined by the application that created the object). You can size the control differently by changing the values of the *height* and *width* parts in the Resize event.

RowColChange Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtRowColChangeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtRowColChangeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtRowColChangeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtRowColChangeS"}

Occurs when the current cell changes to a different cell.

Syntax

Private Sub *object*_**RowColChange (**[*index* **As Integer**, *lastrow* **As String**, *lastcol* **As Integer**]**)**

The RowColChange event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it is in a control array.
lastrow	(For **DBGrid** control) A string expression specifying the previous row position.
lastcol	(For **DBGrid** control) An integer specifying the previous column position.

Remarks

This event occurs whenever the user clicks a cell other than the current cell or when you programmatically change the current cell within a selection using the **Col** and **Row** properties.

The SelChange event also occurs when a user clicks a new cell, but doesn't occur when you programmatically change the selected range without changing the current cell.

For the **DBGrid** control, the position of the current cell is provided by the **Bookmark** and **ColIndex** properties. The previous cell position is specified by *lastrow* and *lastcol*. If you edit data and then move the current cell position to a new row, the update events for the original row are completed before another cell becomes the current cell.

Scroll Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtScrollC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtScrollX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtScrollA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtScrollS"}

Occurs when the scroll box on a **ScrollBar** control, or an object which contains a scrollbar, is repositioned or scrolled horizontally or vertically.

Syntax

Private Sub *dbgrid*_Scroll([*cancel* **As Integer**]**)**
Private Sub *object*_**Scroll( )**

The Scroll event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
cancel	Determines whether the scroll operation succeeds and the **ScrollBar** or **DBGrid** are repainted, as described in Remarks.

Remarks

For a **DBGrid** control, this event occurs when the user scrolls the grid horizontally or vertically but before the grid is repainted to display the results of the scroll operation.

For a **ComboBox** control, this event occurs only when the scrollbars in the dropdown portion of the control are manipulated.

Setting *cancel* to **True** causes the **DBGrid** scroll operation to fail, and no repaint operation occurs. If the **Refresh** method is invoked within this event, the grid is repainted in its new (scrolled) arrangement even if *cancel* is set to **True**. However, in this case, the grid is repainted again because the scroll operation fails and it snaps back to its previous position.

You can use this event to perform calculations or to manipulate controls that must be coordinated with ongoing changes in scroll bars. In contrast, use the Change event when you want an update to occur only once, after a **ScrollBar** control changes.

Note Avoid using a **MsgBox** statement or function in this event.

SelChange Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtSelChangeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtRowColChangeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtSelChangeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtSelChangeS"}

Occurs when the current range changes to a different cell or range of cells.

Syntax

Private Sub DBGrid_SelChange ([*cancel* **As Integer**]**)**
Private Sub *object***_SelChange( )**

The SelChange event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
cancel	Determines whether the selection reverts to its position before the event occurred.

Remarks

The SelChange event occurs whenever a user clicks a cell other than the current cell and as a user drags to select a new range of cells. A user can also select a range of cells by pressing the SHIFT key and using the arrow keys.

You can trigger this event in code for a **DBGrid** control by changing the selected region using the **SelStartCol**, **SelEndCol**, and **SelectedRows( )** properties.

The RowColChange event also occurs when a user clicks a new cell but doesn't occur while a user drags the selection across the **DBGrid** control or when you programmatically change the selection without moving the current cell.

Setting *cancel* to **True** in the **DBGrid** control causes the selection to revert to the cell or range active before the event occurred.

Timer Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtTimerC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtTimerX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtTimerA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtTimerS"}

Occurs when a preset interval for a **Timer** control has elapsed. The interval's frequency is stored in the control's **Interval** property, which specifies the length of time in milliseconds.

Syntax

Private Sub *object*_**Timer(**[*index* **As Integer**]**)**

The Timer event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.

Remarks

Use this event procedure to tell Visual Basic what to do after each **Timer** control interval has elapsed. When you're working with the Timer event:

- The **Interval** property specifies the interval between Timer events in milliseconds.
- Whenever the **Timer** control's **Enabled** property is set to **True** and the **Interval** property is greater than 0, the Timer event waits for the period specified in the **Interval** property.

Unload Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtUnloadC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtUnloadX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtUnloadA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtUnloadS"}

Occurs when a form is about to be removed from the screen. When that form is reloaded, the contents of all its controls are reinitialized. This event is triggered by a user closing the form using the Close command on the Control menu or an **Unload** statement.

Syntax

Private Sub *object*_**Unload(***cancel* **As Integer)**

The Unload event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
cancel	Integer that determines whether the form is removed from the screen. If *cancel* is 0, the form is removed. Setting *cancel* to any nonzero value prevents the form from being removed.

Remarks

Setting *cancel* to any nonzero value prevents the form from being removed, but doesn't stop other events, such as exiting from the Microsoft Windows operating environment. Use the QueryUnload event to stop exiting from Windows.

Use an Unload event procedure to verify that the form should be unloaded or to specify actions that you want to take place when the form is unloaded. You can also include any form-level validation code you may need for closing the form or saving the data in it to a file.

The QueryUnload event occurs before the Unload event. The Unload event occurs before the Terminate event.

The Unload event can be caused by using the **Unload** statement, or by the user choosing the Close command on a form's Control menu, exiting the application with the End Task button on the Windows Task List, closing the MDI form for which the current form is a child form, or exiting the Microsoft Windows operating environment while the application is running.

Activate, Deactivate Events Example

This example updates the status bar text to display the caption of the active form. To try this example**,** create a **Form** object (Form1) and a new **MDIForm** object (MDIForm1). On MDIForm1, draw a **PictureBox** control containing a **Label** control. On Form1, set the **MDIChild** property to **True**. Paste the MDIForm_Load event procedure code into the Declarations section of the **MDIForm** object. Paste the Form_Activate event procedure code into the Declarations section of the MDI child form, and then press F5.

```
Private Sub MDIForm_Load ()
   Form1.Caption = "Form #1"       ' Set caption of Form1.
   Dim NewForm As New Form1        ' Create a new child form.
   Load NewForm
   NewForm.Caption = "Form #2"     ' Set caption of new form.
   NewForm.Show                    ' Display the new form.
End Sub

Private Sub Form_Activate ()
      ' Set status bar text.
   MDIForm1.Label1.Caption = "Current form: " & Me.Caption
End Sub
```

Change Event Example

This example displays the numeric setting of a horizontal scroll bar's **Value** property in a **TextBox** control. To try this example, create a form with a **TextBox** control and an **HScrollBar** control and then paste the code into the Declarations section of a form that contains a horizontal scroll bar (**HScrollBar** control) and a **TextBox** control. Press F5 and click the horizontal scroll bar.

```
Private Sub Form_Load ()
   HScroll1.Min = 0                ' Set Minimum.
   HScroll1.Max = 1000             ' Set Maximum.
   HScroll1.LargeChange = 100      ' Set LargeChange.
   HScroll1.SmallChange = 1        ' Set SmallChange.
End Sub

Private Sub HScroll1_Change ()
   Text1.Text = HScroll1.Value
End Sub
```

Click Event Example

In this example, each time a **PictureBox** control is clicked it moves diagonally across a form. To try this example, paste the code into the Declarations section of a form that contains a **PictureBox** control positioned at the lower-left corner of the form, and then press F5 and click the **PictureBox**.

```
Private Sub Picture1_Click ()
   Picture1.Move Picture1.Left + 750, Picture1.Top - 550
End Sub
```

DragDrop Event Example

This example demonstrates the visual effect of dropping a **PictureBox** control onto another **PictureBox** control. To try this example, paste the code into the Declarations section of a form that contains three **PictureBox** controls. Set the **DragMode** property for Picture1 and Picture2 to 1 (Automatic). Use the **Picture** property to assign bitmaps to Picture1 and Picture2, and then press F5 and drag Picture1 or Picture2 over Picture3.

```
Private Sub Picture3_DragDrop (Source As Control, X as Single, Y As Single)
   If TypeOf Source Is PictureBox Then
      ' Set Picture3 bitmap to same as source control.
      Picture3.Picture = Source.Picture
   End If
End Sub
```

DragOver Event Example

This example demonstrates one way to indicate a valid drop target. The pointer changes from the default arrow to a special icon when a **TextBox** control is dragged over a **PictureBox** control. The pointer returns to the default when the source is dragged elsewhere. To try this example, paste the code into the Declarations section of a form that contains a small **TextBox** and a **PictureBox**. Set the **TextBox** control's **DragMode** property to 1, and then press F5 and drag the **TextBox** over the **PictureBox**.

```
Private Sub Picture1_DragOver (Source As Control, X As Single, Y As Single,
State As Integer)
   Select Case State
      Case vbEnter
   ' Load icon.
         Source.DragIcon = LoadPicture("ICONS\ARROWS\POINT03.ICO")
      Case vbLeave
         Source.DragIcon = LoadPicture() ' Unload icon.
   End Select
End Sub

Private Sub Picture1_DragDrop (Source As Control, X As Single, Y As Single)
   Source.DragIcon = LoadPicture() ' Unload icon.
End Sub
```

DropDown Event Example

This example updates a **ComboBox** control based on the user's selection in an option button group. To try this example, paste the code into the Declarations section of a form that contains a **ComboBox** control and two **OptionButton** controls. Set the **Name** property of both **OptionButton** controls to OptionGroup, and then press F5 and click the **OptionButton** controls. The **ComboBox** control reflects different carriers depending on the **OptionButton** selected.

```
Private Sub Form_Load ()
   Combo1.Text = ""  ' Clear combo box.
End Sub

Private Sub Combo1_DropDown ()
   Combo1.Clear            ' Delete existing items.
   If OptionGroup(0).Value = True Then
      Combo1.AddItem "Gray Goose Express", 0
      Combo1.AddItem "Wild Fargo Carriers", 1
   Else
      Combo1.AddItem "Summit Technologies Overnight"
   End If
End Sub
```

GotFocus Event Example

This example displays a status bar message when a button in an **OptionButton** group gets the focus. To try this example, paste the code into the Declarations section of a form that contains two **OptionButton** controls and a **Label** control. Set the **Name** property for both **OptionButton** controls to OptionGroup, and then press F5 and click the **OptionButton** controls.

```
Private Sub Form_Load ()
   Label1.AutoSize = True
End Sub

Private Sub OptionGroup_GotFocus (Index As Integer)
   Select Case Index
      Case 0
         Label1.Caption = "Option 1 has the focus."
      Case 1
         Label1.Caption = "Option 2 has the focus."
   End Select
End Sub

Private Sub OptionGroup_LostFocus (Index As Integer)
   Label1.Caption = ""
End Sub
```

KeyDown, KeyUp Events Example

This example demonstrates a generic keyboard handler that responds to the F2 key and to all the associated ALT, SHIFT, and CTRL key combinations. The key constants are listed in the Visual Basic (VB) object library in the Object Browser. To try this example, paste the code into the Declarations section of a form that contains a **TextBox** control, and then press F5 and press F2 with various combinations of the ALT, SHIFT, and CTRL keys.

```
Private Sub Text1_KeyDown (KeyCode As Integer, Shift As Integer)
   Dim ShiftDown, AltDown, CtrlDown, Txt
   ShiftDown = (Shift And vbShiftMask) > 0
   AltDown = (Shift And vbAltMask) > 0
   CtrlDown = (Shift And vbCtrlMask) > 0
   If KeyCode = vbKeyF2 Then ' Display key combinations.
      If ShiftDown And CtrlDown And AltDown Then
         Txt = "SHIFT+CTRL+ALT+F2."
      ElseIf ShiftDown And AltDown Then
         Txt = "SHIFT+ALT+F2."
      ElseIf ShiftDown And CtrlDown Then
         Txt = "SHIFT+CTRL+F2."
      ElseIf CtrlDown And AltDown Then
         Txt = "CTRL+ALT+F2."
      ElseIf ShiftDown Then
         Txt = "SHIFT+F2."
      ElseIf CtrlDown Then
      Txt = "CTRL+F2."
      ElseIf AltDown Then
         Txt = "ALT+F2."
      ElseIf SHIFT = 0 Then
         Txt = "F2."
      End If
   Text1.Text = "You pressed " & Txt
   End If
End Sub
```

KeyPress Event Example

This example converts text entered into a **TextBox** control to uppercase. To try this example, paste the code into the Declarations section of a form that contains a **TextBox**, and then press F5 and enter something into the **TextBox**.

```
Private Sub Text1_KeyPress (KeyAscii As Integer)
   Char = Chr(KeyAscii)
   KeyAscii = Asc(UCase(Char))
End Sub
```

LinkError Event Example

This example is attached to a **TextBox** control, MyTextBox, that handles selected errors. The procedure displays a message (adapted from the error list in the LinkError event topic) based on the error number passed as the argument `LinkErr`. You can adapt this code to a source form by substituting `Form_LinkError` for `MyTextBox_LinkError`. This example is for illustration only.

```
Private Sub MyTextBox_LinkError (LinkErr As Integer)
Dim Msg
Select Case LinkErr
   Case 1
      Msg = "Data in wrong format."
   Case 11
      Msg = "Out of memory for DDE."
End Select
MsgBox Msg, vbExclamation, "MyTextBox"
End Sub
```

LinkExecute Event Example

This example defines a set of commands for destinations to use in DDE conversations to which your application will respond. This example is for illustration only.

```
Private Sub Form_LinkExecute (CmdStr As String, Cancel As Integer)
   Cancel = False
   Select Case LCase(CmdStr)
   Case "{big}"
      WindowState = 2    ' Maximize window.
   Case "{little}"
      WindowState = 1    ' Minimize window.
   Case "{hide}"
      Visible = False    ' Hide form.
   Case "{view}"
      Visible = True     ' Display form.
   Case Else
      Cancel = True      ' Execute not allowed.
   End Select
End Sub
```

LinkNotify Event Example

This example is attached to a **PictureBox** control, Picture1, that has its **LinkTopic** and **LinkItem** properties set to specify a graphic in the source, and its **LinkMode** property set to 3 (Notify). When the source changes this data, the procedure updates the **PictureBox** control immediately only if the **PictureBox** is on the active form; otherwise, it sets a flag variable. This example is for illustration only.

```
Private Sub Picture1_LinkNotify ()
   If Screen.ActiveForm Is Me Then
   Picture1.LinkRequest    ' Picture is on active form, so update.
   Else
      NewDataFlag = True   ' Assumed to be a module-level variable.
   End If
End Sub
```

Load Event Example

This example loads items into a **ComboBox** control when a form is loaded. To try this example, paste the code into the Declarations section of a form that contains a **ComboBox**, and then press F5.

```
Private Sub Form_Load ()
   Combo1.AddItem "Mozart"    ' Add items to list.
   Combo1.AddItem "Beethoven"
   Combo1.AddItem "Rock 'n Roll"
   Combo1.AddItem "Reggae"
   Combo1.ListIndex = 2       ' Set default selection.
End Sub
```

LostFocus Event Example

This example changes the color of a **TextBox** control when it receives or loses the focus (selected with the mouse or TAB key) and displays the appropriate text in the **Label** control. To try this example, paste the code into the Declarations section of a form that contains two **TextBox** controls and a **Label** control, and then press F5 and move the focus between Text1 and Text2.

```
Private Sub Text1_GotFocus ()
   ' Show focus with red.
   Text1.BackColor = RGB(255, 0, 0)
   Label1.Caption = "Text1 has the focus."
End Sub

Private Sub Text1_LostFocus ()
   ' Show loss of focus with blue.
   Text1.BackColor = RGB(0, 0, 255)
   Label1.Caption = "Text1 doesn't have the focus."
End Sub
```

MouseDown, MouseUp Events Example

This example demonstrates a simple paint application. The MouseDown event procedure works with a related MouseMove event procedure to enable painting when any mouse button is pressed and dragged. The MouseUp event procedure disables painting. To try this example, paste the code into the Declarations section of a form; and then press F5, click the form, and move the mouse while the mouse button is pressed.

```
Dim PaintNow As Boolean
Private Sub Form_MouseDown (Button As Integer, Shift As Integer, X As
Single, Y As Single)
   PaintNow = True   ' Enable painting.
End Sub

Private Sub Form_MouseUp (Button As Integer, Shift As Integer, X As Single,
Y As Single)
   PaintNow = False  ' Disable painting.
End Sub

Private Sub Form_MouseMove (Button As Integer, Shift As Integer, X As
Single, Y As Single)
   If PaintNow Then
      PSet (X, Y)    ' Draw a point.
   End If
End Sub

Private Sub Form_Load ()
   DrawWidth = 10    ' Use wider brush.
   ForeColor = RGB(0, 0, 255)      ' Set drawing color.
End Sub
```

MouseMove Event Example

This example demonstrates a simple paint application. The MouseDown event procedure works with a related MouseMove event procedure to enable painting when any mouse button is pressed. The MouseUp event procedure disables painting. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form and move the mouse while the mouse button is pressed.

```
Dim PaintNow As Boolean ' Declare variable.
Private Sub Form_MouseDown (Button As Integer, Shift As Integer, X As
Single, Y As Single)
   PaintNow = True   ' Brush on.
End Sub

Private Sub Form_MouseUp (Button As Integer, X As Single, Y As Single)
   PaintNow = False  ' Turn off painting.
End Sub

Private Sub Form_MouseMove (Button As Integer, Shift As Integer, X As
Single, Y As Single)
   If PaintNow Then
      PSet (X, Y)    ' Draw a point.
   End If
End Sub

Private Sub Form_Load ()
   DrawWidth = 10    ' Use wider brush.
   ForeColor = RGB(0, 0, 255)      ' Set drawing color.
End Sub
```

Paint Event Example

This example draws a diamond that intersects the midpoint of each side of a form and adjusts automatically as the form is resized. To try this example, paste the code into the Declarations section of a form, and then press F5 and resize the form.

```
Private Sub Form_Paint ()
   Dim HalfX, HalfY  ' Declare variables.
   HalfX = ScaleLeft + ScaleWidth / 2    ' Set to one-half of width.
   HalfY = ScaleTop + ScaleHeight / 2    ' Set to one-half of height.
   ' Draw a diamond.
   Line (ScaleLeft, HalfY) - (HalfX, ScaleTop)
   Line -(ScaleWidth + ScaleLeft, HalfY)
   Line -(HalfX, ScaleHeight + ScaleTop)
   Line -(ScaleLeft, HalfY)
End Sub

Private Sub Form_Resize
   Refresh
End Sub
```

PathChange Event Example

This example demonstrates how to update a **Label** control to reflect the current path for a **FileListBox** control. Double-clicking a directory name displays a list of that directory's files in the **FileListBox**; it also displays the directory's complete path in the **Label** control. To try this example, paste the code into the Declarations section of a form that contains a **Label** control, a **DirListBox** control, and a **FileListBox** control, and then press F5. Double-click a directory to change the path.

```
Private Sub File1_PathChange ()
   Label1.Caption = "Path: " & Dir1.Path ' Show path in Label.
End Sub

Private Sub Dir1_Change ()
   File1.Path = Dir1.Path                ' Set file path.
End Sub

Private Sub Form_Load ()
   Label1.Caption = "Path: " & Dir1.Path ' Show path in Label.
End Sub
```

PatternChange Event Example

This example updates a **FileListBox** control with files matching the pattern entered in a **TextBox** control. If a full path is entered into the **TextBox**, such as C:\BIN*.EXE, the text is automatically parsed into path and pattern components. To try this example, paste the code into the Declarations section of a form that contains a **TextBox** control, a **Label** control, a **FileListBox** control, and a **CommandButton** control, and then press F5 and enter a valid file pattern in the **TextBox**.

```
Private Sub Form_Load ()
   Command1.Default = True          ' Set Default property.
   Command1.Caption = "OK"          ' Set Caption.
End Sub

Private Sub Command1_Click ()' OK button clicked.
   ' Text is parsed into path and pattern components.
   File1.FileName = Text1.Text
   Label1.Caption = "Path: " & File1.Path
End Sub

Private Sub File1_PatternChange ()
   Text1.Text = File1.Pattern       ' Set text to new pattern.
End Sub
```

QueryUnload Event Example

This example uses an **MDIForm** object containing two MDI child forms. When you choose the Close command from the Control menu to close a form, a different message is displayed than if you choose the Exit command from the File menu. To try this example, create an **MDIForm**, and then use the Menu Editor to create a File menu containing an Exit command named FileExit. Make sure that this menu item is enabled. On Form1, set the **MDIChild** property to **True**. Paste the code into the Declarations sections of the respective forms, and then press F5 to run the program.

```
' Paste into Declarations section of MDIForm1.
Private Sub MDIForm_Load ()
   Dim NewForm As New Form1  ' New instance of Form1.
   NewForm.Caption = "Form2" ' Set caption and show.
End Sub

Private Sub FileExit_Click ()
   Unload MDIForm1   ' Exit the application.
End Sub

Private Sub MDIForm_QueryUnload (Cancel As Integer, UnloadMode As Integer)
   Dim Msg  ' Declare variable.
   ' Set the message text.
   Msg = "Do you really want to exit the application?"
   ' If user clicks the No button, stop QueryUnload.
   If MsgBox(Msg, vbQuestion + vbYesNo, Me.Caption) = vbNo Then Cancel =
True
End Sub

' Paste into Declarations section of Form1.
Private Sub Form_QueryUnload (Cancel As Integer, UnloadMode As Integer)
   Dim Msg  ' Declare variable.
   If UnloadMode > 0 Then
      ' If exiting the application.
      Msg = "Do you really want to exit the application?"
   Else
      ' If just closing the form.
      Msg = "Do you really want to close the form?"
   End If
   ' If user clicks the No button, stop QueryUnload.
   If MsgBox(Msg, vbQuestion + vbYesNo, Me.Caption) = vbNo Then Cancel =
True
End Sub
```

Resize Event Example

This example automatically resizes a **TextBox** control to fill the form whenever the form is resized. To try this example, paste the code into the Declarations section of a form that contains a **TextBox**. Set the **TextBox** control's **MultiLine** property to **True**, its **ScrollBars** property to 3, and its **BorderStyle** property to 0, and then press F5 and resize the form.

```
Private Sub Form_Load ()
   Text1.Text = ""    ' Clear text.
End Sub

Private Sub Form_Resize ()
   Text1.Move 0,0, ScaleWidth, ScaleHeight
End Sub
```

RowColChange, SelChange Events Example; Cols, Rows Properties Example

This example displays the location of the current cell and the range of the selection as a user selects cells or ranges. When selecting a range, the current cell doesn't change. Select a range, and then click the form to move the current cell around the perimeter of the selection. Notice that the selected range doesn't change.

To try this example, create a new project, use the Components dialog box to add a **Grid** control to the toolbox (from the Project menu, choose Components, and then select Microsoft Grid Control), and then draw a **Grid** control and two **Label** controls. Copy the code into the Declarations section, and then press F5 to run the program.

```
Private Sub Form_Load ()
   Grid1.Cols = 6 ' Set columns and rows.
   Grid1.Rows = 7
End Sub

Private Sub Grid1_RowColChange ()
   Msg = "Current Cell: " & Chr(64 + Grid1.Col)
   Mst = Msg & Grid1.Row
   Label1.Caption = Msg
End Sub

Private Sub Grid1_SelChange ()
   Msg = "Selection: " & Chr(64 + Grid1.SelStartCol)
   Msg = Msg & Grid1.SelStartRow
   Msg = Msg & ":" & Chr(64 + Grid1.SelEndCol)
   Msg = Msg & Grid1.SelEndRow
   Label2.Caption = Msg
End Sub

Private Sub Form_Click ()
   ' This procedure moves the current cell around
   ' the perimeter of the selected range
   ' of cells with each click on the form.
   Dim GR, GC As Integer
   If Grid1.Row = Grid1.SelStartRow Then
      If Grid1.Col = Grid1.SelEndCol Then
         GR = 1: GC = 0
      Else
         GR = 0: GC = 1
      End If
   ElseIf Grid1.Row = Grid1.SelEndRow Then
      If Grid1.Col = Grid1.SelStartCol Then
         GR = -1: GC = 0
      Else
         GR = 0: GC = -1
      End If
   Else
      If Grid1.Col = Grid1.SelStartCol Then
         GR = -1: GC = 0
      Else
         GR = 1: GC = 0
      End If
   End If
   Grid1.Row = Grid1.Row + GR
```

```
   Grid1.Col = Grid1.Col + GC
End Sub
```

Scroll Event Example

This example changes the size of a **Shape** control to correspond to the value of a horizontal scroll bar (**HScrollBar**) as you drag the scroll box on the scroll bar. To try this example, paste the code into the Declarations section of a form that contains a **Shape** control, a **Label** control, and an **HScrollBar** control. Set the **Index** property for the **Shape** control to 0 to create a control array. Then press F5 and move the scroll bar.

```
Private Sub Form_Load ()
   ' Move and size the first Shape control.
   Shape1(0).Move HScroll1.Left, HScroll1.Top * 1.5, HScroll1.Width,
HScroll1.Height
   Label1.Caption = ""  ' Set the Label caption.
   Load Shape1(1) ' Create the second Shape.
   ' Move and size the second Shape control.
   Shape1(1).Move Shape1(0).Left, Shape1(0).Top, 1, Shape1(0).Height
   Shape1(1).BackStyle = 1 ' Set BackStyle to Opaque.
   Shape1(1).Visible = True  ' Display the second Shape.
   HScroll1.Min = 1  ' Set values of the scroll bar.
   HScroll1.Max = HScroll1.Width
End Sub

Private Sub HScroll1_Change ()
   Label1.Caption = "Changed"      ' Display message after change.
End Sub

Private Sub HScroll1_Scroll ()
   Shape1(1).BackColor = &HFF0000  ' Set Shape color to Blue.
   Label1.Caption = "Changing"     ' Display message while scrolling.
   Shape1(1).Width = HScroll1.Value ' Size Shape to Scroll Value.
End Sub
```

Timer Event Example

This example demonstrates a digital clock. To try this example, paste the code into the Declarations section of a form that contains a **Label** control and a **Timer** control, and then press F5.

```
Private Sub Form_Load ()
   Timer1.Interval = 1000  ' Set Timer interval.
End Sub

Private Sub Timer1_Timer ()
   Label1.Caption = Time   ' Update time display.
End Sub
```

This example moves a **PictureBox** control across a form. To try this example, paste the code into the Declarations section of a form that contains a **Timer** control and a **PictureBox** control, and then press F5. For a better visual effect you can assign a bitmap to the **PictureBox** using the **Picture** property.

```
Dim DeltaX, DeltaY As Integer' Declare variables.
Private Sub Timer1_Timer ()
   Picture1.Move Picture1.Left + DeltaX, Picture1.Top + DeltaY
   If Picture1.Left < ScaleLeft Then DeltaX = 100
   If Picture1.Left + Picture1.Width > ScaleWidth + ScaleLeft Then
      DeltaX = -100
   End If
   If Picture1.Top < ScaleTop Then DeltaY = 100
   If Picture1.Top + Picture1.Height > ScaleHeight + ScaleTop Then
      DeltaY = -100
   End If
End Sub

Private Sub Form_Load ()
   Timer1.Interval = 1000  ' Set Interval.
   DeltaX = 100   ' Initialize variables.
   DeltaY = 100
End Sub
```

Unload Event Example

This example demonstrates a simple procedure to close a form while prompting the user with various message boxes. In an actual application, you can add calls to general purpose **Sub** procedures that emulate the processing of the Exit, Save, and Save As commands on the File menu in Visual Basic. To try this example, paste the code into the Declarations section of a form, and then press F5. Once the form is displayed, press ALT+F4 to close the form.

```
Private Sub Form_Unload (Cancel As Integer)
   Dim Msg, Response ' Declare variables.
   Msg = "Save Data before closing?"
   Response = MsgBox(Msg, vbQuestion + vbYesNoCancel, "Save Dialog")
   Select Case Response
      Case vbCancel  ' Don't allow close.
         Cancel = -1
         Msg = "Command has been canceled."
      Case vbYes
      ' Enter code to save data here.
         Msg = "Data saved."'
      Case vbNo
         Msg = "Data not saved."
   End Select
   MsgBox Msg, vbOKOnly, "Confirm" ' Display message.
End Sub
```

ItemCheck Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtItemCheckEventC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtItemCheckEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtItemCheckEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtItemCheckEventS"}

Occurs when a **ListBox** control **Style** property is set to 1 (checkboxes) and an item's checkbox in the **ListBox** control is selected or cleared.

Syntax

Private Sub *object*_**ItemCheck(**[*index* **As Integer**]**)**

The ItemCheck event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies the item in the listbox which was clicked.

Remarks

Note The ItemCheck event does not occur when a list item is only highlighted; rather, it occurs when the check box of the list item is selected or cleared.

The ItemCheck event can also occur programmatically whenever an element in Selected array of the **ListBox** is changed (and its **Style** property is set to 1.)

The ItemCheck event occurs before the Click event.

Property Pages

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproListOfPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproListOfPropertiesX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproListOfPropertiesA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproListOfPropertiesS"}

To get Help for the properties contained in a control's Property Page dialog box, click the button below corresponding to the control in which you are interested.Then click the property name to get Help for that property.

{button ,JI(`',`propAnimation')} **Animation** control

{button ,JI(`',`propCommonDialog')} **CommonDialog** control

{button ,JI(`',`propDBCombo')} **DBCombo** control

{button ,JI(`',`propDBList')} **DBList** control

{button ,JI(`',`propImageList')} **ImageList** control

{button ,JI(`',`propListView')} **ListView** control

{button ,JI(`',`propMAPIMessages')} **MAPIMessages** control

{button ,JI(`',`propMAPISession')} **MAPISession** control

{button ,JI(`',`propMasked')} **MaskedEdit** control

{button ,JI(`',`propMSChart')} **MSChart** control

{button ,JI(`',`propMSComm')} **MSComm** control

{button ,JI(`',`propMSFlexGrid')} **MSFlexGrid** control

{button ,JI(`',`propMMMCI')} **Multimedia MCI** control

{button ,JI(`',`propPicclip')} **PictureClip** control

{button ,JI(`',`propProgressBar')} **ProgressBar** control

{button ,JI(`',`propRTFBox')} **RichTextBox** control

{button ,JI(`',`propSlider')} **Slider** control

{button ,JI(`',`propSSTab')} **SSTab** control

{button ,JI(`',`propStatusBar')} **StatusBar** control

{button ,JI(`',`propTabStrip')} **TabStrip** control

{button ,JI(`',`propToolBar')} **ToolBar** control

{button ,JI(`',`propTreeView')} **TreeView** control

{button ,JI(`',`propUpDown')} **UpDown** control

{button ,JI(`',`propWinSock')} **Winsock** control

Property Pages

Animation control

AutoPlay	AutoPlay
BackStyle	BackStyle
Center	Center
OLEDropMode	OLEDropMode

Common Dialog control

CancelError	CancelError
Color	Color
Copies	Copies

DefaultExt	DefaultExt
DialogTitle	DialogTitle
FileName	FileName
Filter	Filter
FilterIndex	FilterIndex
Flags	Flags
FontName	FontName
FontSize	FontSize
FromPage	FromPage
HelpCommand	HelpCommand
HelpContext	HelpContext
HelpFile	HelpFile
HelpKey	HelpKey
InitDir	InitDir
Max	Max
MaxFileSize	MaxFileSize
Min	Min
PrinterDefault	PrinterDefault
ToPage	ToPage

DBCombo control

Appearance	Appearance
Enabled	Enabled
IntegralHeight	IntegralHeight
Locked	Locked
MatchEntry	MatchEntry
MousePointer	MousePointer
OLEDragMode	OLEDragMode
OLEDropMode	OLEDropMode
Style	Style

DBList control

Appearance	Appearance
Enabled	Enabled
IntegralHeight	IntegralHeight
Locked	Locked
MatchEntry	MatchEntry
MousePointer	MousePointer
OLEDragMode	OLEDragMode
OLEDropMode	OLEDropMode

ImageList control

Height	Height
Image	Image

Image Count	Image Count
Index	Index
Key	Key
Tag	Tag
UseMaskColor	UseMaskColor
Width	Width

ListView control

Alignment	Alignment
Appearance	Appearance
Arrange	Arrange
BorderStyle	BorderStyle
Enabled	Enabled pro
HideColumnHeaders	HideColumnHeaders
HideSelection	HideSelection
ImageList	ImageList
Index	Index
Key	Key
LabelEdit	LabelEdit
LabelWrap	LabelWrap
MousePointer	MousePointer
MultiSelect	MultiSelect
OLEDragMode	OLEDragMode
OLEDropMode	OLEDropMode
Sorted	Sorted
SortKey	SortKey
SortOrder	SortOrder
Tag	Tag
Text	Text
View	View
Width	Width

MAPIMessages control

AddressCaption	AddressCaption
AddressEditFieldCount	AddressEditFieldCount
AddressLabel	AddressLabel
AddressModifiable	AddressModifiable
AddressResolveUI	AddressResolveUI
FetchMsgType	FetchMsgType
FetchSorted	FetchSorted
FetchUnreadOnly	FetchUnreadOnly

MAPISession control

DownloadMail	DownloadMail

LogonUI	LogonUI
NewSession	NewSession
Password	Password
UserName	UserName

MaskedEdit control

AllowPrompt	AllowPrompt
AutoTab	AutoTab
BorderStyle	BorderStyle
ClipMode	ClipMode
Enabled	Enabled
Format	Format
HideSelection	HideSelection
Mask	Mask
MaxLength	MaxLength
MousePointer	MousePointer
OLEDragMode	OLEDragMode
OLEDropMode	OLEDropMode
PromptChar	PromptChar
PromptInclude	PromptInclude

MSChart control

2D	2D
3D	3D
Alignment	Alignment
Automatic scaling	Automatic scaling
Axis	Axis
Color	Color
Exclude series	Exclude series
Font style	Font style
Font	Font
Hide series	Hide series
Major divisions	Major divisions
Maximum	Maximum
Mean	Mean
Minimum	Minimum
Minor divisions	Minor divisions
Orientation	Orientation
Pattern color	Pattern color
Pattern	Pattern
Plot on 2nd Y axis	Plot on 2nd Y axis
Property Name	Property Name
Regression	Regression
Series in rows	Series in rows
Series type	Series type

Series type	Series type
Series	Series
Series	Series
Shadow	Shadow
Show legend	Show legend
Show markers	Show markers
Show markers	Show markers
Show scale	Show scale
Size	Size
Stack series	Stack series
Standard deviation	Standard deviation
Strikeout	Strikeout
Style	Style
Style	Style
Style	Style
Style	Style
Style	Style
Text	Text
Underline	Underline
Width	Width
Width	Width
Width	Width
Width	Width
Width	Width

MSComm control

CommPort	CommPort
DTREnable	DTREnable
EOFEnable	EOFEnable
Handshaking	Handshaking
InBufferSize	InBufferSize
InputLen	InputLen
NullDiscard	NullDiscard
OutBufferSize	OutBufferSize
ParityReplace	ParityReplace
Rthreshold	RThreshold
RTSEnable	RTSEnable
Settings	Settings
Sthreshold	SThreshold

MSFlexGrid control

AllowBigSelection	AllowBigSelection
AllowUserResizing	AllowUserResizing
Cols	Cols

FillStyle	FillStyle
FixedCols	FixedCols
FixedRows	FixedRows
FocusRect	FocusRect
Font	Font
Format	Format
GridLines	GridLines
GridLinesFixed	GridLinesFixed
HighLight	HighLight
MergeCells	MergeCells
MousePointer	MousePointer
PictureType	PictureType
RowHeightMin	RowHeightMin
Rows	Rows
ScrollBars	ScrollBars
SelectionMode	SelectionMode
TextStyle	TextStyle
TextStyleFixed	TextStyleFixed
WordWrap	WordWrap

Multimedia MCI control

AutoEnable	AutoEnable
BackEnabled	BackEnabled
BackVisible	BackVisible
BorderStyle	BorderStyle
DeviceType	DeviceType
EjectEnabled	EjectEnabled
EjectVisible	EjectVisible
Enabled	Enabled
FileName	FileName
Frames	Frames
MousePointer	MousePointer
NextEnabled	NextEnabled
NextVisible	NextVisible
OLEDropMode	OLEDropMode
Orientation	Orientation
PauseEnabled	PauseEnabled
PauseVisible	PauseVisible
PlayEnabled	PlayEnabled
PlayVisible	PlayVisible
PrevEnabled	PrevEnabled
PrevVisible	PrevVisible
RecordEnabled	RecordEnabled
RecordMode	RecordMode

RecordVisible	RecordVisible
Shareable	Shareable
Silent	Silent
StepEnabled	StepEnabled
StepVisible	StepVisible
StopEnabled	StopEnabled
StopVisible	StopVisible
UpdateInterval	UpdateInterval

PictureClip control

Cols	Cols
Rows	Rows

ProgressBar control

Appearance	Appearance
BorderStyle	BorderStyle
Enabled	Enabled
Max	Max
Min	Min
MousePointer	MousePointer
OLEDropMode	OLEDropMode

RichTextBox control

Appearance	Appearance
AutoVerbMenu	AutoVerbMenu
BorderStyle	BorderStyle
BulletIndent	BulletIndent
DisableNoScroll	DisableNoScroll
Enabled	Enabled
FileName [load from]	FileName [load from]
HideSelection	HideSelection
Locked	Locked
MaxLength	MaxLength
MousePointer	MousePointer
MultiLine	MultiLine
OLEDragMode	OLEDragMode
OLEDropMode	OLEDropMode
RightMargin	RightMargin
ScrollBars	ScrollBars

Slider control

Enabled	Enabled
LargeChange	LargeChange
Max	Max

Min	Min
MousePointer	MousePointer
OLEDropMode	OLEDropMode
Orientation	Orientation
SelectRange	SelectRange
SelLength	SelLength
SelStart	SelStart
SmallChange	SmallChange
TickFrequency	TickFrequency
TickStyle	TickStyle

SSTab control

Current Tab	CurrentTab
Enabled	Enabled
MousePointer	MousePointer
OLEDropMode	OLEDropMode
Orientation	Orientation
ShowFocusRect	ShowFocusRect
Style	Style
Tab Count	Tab Count
TabCaption	TabCaption
TabHeight	TabHeight
TabMaxWidth	TabMaxWidth
TabsPerRow	TabsPerRow
WordWrap	WordWrap

StatusBar control

Actual Width	Actual Width
Alignment	Alignment
AutoSize	AutoSize
Bevel	Bevel
Enabled	Enabled
Index	Index
Key	Key
Minimum Width	Minimum Width
MousePointer	MousePointer
OLEDropMode	OLEDropMode
Picture	Picture
ShowTips	ShowTips
SimpleText	SimpleText
Style	Style
Tag	Tag
Text	Text
ToolTipText	ToolTipText

TabStrip control

Caption	Caption
Enabled	Enabled
Image	Image
ImageList	ImageList
Index	Index
Key	Key
MousePointer	MousePointer
MultiRow	MultiRow
OLEDropMode	OLEDropMode
ShowTips	ShowTips
Style	Style
TabFixedHeight	TabFixedHeight
TabFixedWidth	TabFixedWidth
TabWidthStyle	TabWidthStyle
Tag	Tag
ToolTipText	ToolTipText

TreeView control

Appearance	Appearance
BorderStyle	BorderStyle
Enabled	Enabled
HideSelection	HideSelection
ImageList	ImageList
Indentation	Indentation
LabelEdit	LabelEdit
LineStyle	LineStyle
MousePointer	MousePointer
OLEDragMode	OLEDragMode
OLEDropMode	OLEDropMode
PathSeparator	PathSeparator
Sorted	Sorted
Style	Style

ToolBar control

AllowCustomize	AllowCustomize
Appearance	Appearance
BorderStyle	BorderStyle
ButtonHeight	ButtonHeight
ButtonWidth	ButtonWidth
Caption	Caption
Description	Description
Enabled	Enabled

HelpContextID	HelpContextID
HelpFile	HelpFile
Image	Image
ImageList	ImageList
Index	Index
Key	Key
MixedState	MixedState
MousePointer	MousePointer
OLEDropMode	OLEDropMode
ShowTips	ShowTips
Style	Style
Tag	Tag
ToolTipText	ToolTipText
Value	Value
Visible	Visible
Width	Width
Wrappable	Wrappable

UpDown control

Alignment	Alignment
AutoBuddy	AutoBuddy
BuddyControl	BuddyControl
BuddyProperty	BuddyProperty
Increment	Increment
Max	Max
Min	Min
OLEDropMode	OLEDropMode
Orientation	Orientation
SyncBuddy	SyncBuddy
Value	Value
Wrap	Wrap

Winsock control

LocalPort	LocalPort
Protocol	Protocol
RemoteHost	RemoteHost
RemotePort	RemotePort

Convert a database

The database file you selected isn't a replicable file. To make a replica of your database, you must first convert it to a Design Master. Converting your database increases the size of your file because new, hidden system fields and tables are added to your database.

Before you convert your database, remove the database password. Setting user permissions to access the database doesn't interfere with synchronization.

Make a backup copy

The backup copy is to be used only in emergency cases. Any copy made from the backup copy of the database, or any replica made from a converted backup copy, will not be able to synchronize with the existing members of the replica set.

Designate the Design Master

If you want to change the design of your database while working on the Briefcase copy of the file, make the Briefcase copy the Design Master. It is important to remember, however, that no one will be able to make changes to the design of the replica remaining on the desktop.

Keyword Not Found

The keyword you've selected can't be found in Visual Basic Help. You may have misspelled the keyword, selected too much or too little text, or asked for help on a word that is not a valid Visual Basic keyword.

The easiest way to get help on a specific keyword is to position the insertion point anywhere within the keyword you want help on and press F1. You do not need to select the keyword. In fact, if you select only a portion of the keyword, or more than a single word, Help will not find what you're looking for.

To use the built-in Help search dialog box, press the "Help Topics" button on the toolbar.

Screen Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjScreenC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjScreenX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjScreenP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjScreenM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjScreenE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjScreenS"}

Manipulates forms according to their placement on the screen and controls the mouse pointer outside your application's forms at run time. The **Screen** object is accessed with the keyword **Screen**.

Syntax

Screen

Remarks

The **Screen** object is the entire Windows desktop. Using the **Screen** object, you can set the **MousePointer** property of the **Screen** object to the hourglass pointer while a modal form is displayed.

Refresh Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthRefreshC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthRefreshX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthRefreshA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthRefreshS"}

Forces a complete repaint of a form or control.

Syntax

object.**Refresh**

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

Remarks

Use the **Refresh** method when you want to:

- Completely display one form while another form loads.
- Update the contents of a file-system list box, such as a **FileListBox** control.
- Update the data structures of a **Data** control.

Refresh can't be used on MDI forms , but can be used on MDI child forms. You can't use **Refresh** on **Menu** or **Timer** controls.

Generally, painting a form or control is handled automatically while no events are occurring. However, there may be situations where you want the form or control updated immediately. For example, if you use a file list box, a directory list box, or a drive list box to show the current status of the directory structure, you can use **Refresh** to update the list whenever a change is made to the directory structure.

You can use the **Refresh** method on a **Data** control to open or reopen the database (if the **DatabaseName**, **ReadOnly**, **Exclusive**, or **Connect** property settings have changed) and rebuild the dynaset in the control's **Recordset** property.

Refresh Method Example

This example uses the **Refresh** method to update a **FileListBox** control as test files are created. To try this example, paste the code into the Declarations section of a form with a **FileListBox** control named File1, and then run the example and click the form.

```
Private Sub Form_Click ()
   ' Declare variables.
   Dim FilName, Msg as String, I as Integer
   File1.Pattern = "TestFile.*"    ' Set file pattern.
   For I = 1 To 8 ' Do eight times.
      FilName = "TESTFILE." & I
      ' Create empty file.
      Open FilName For Output As FreeFile
      File1.Refresh  ' Refresh file list box.
      Close ' Close file.
   Next I
   Msg = "Choose OK to remove the created test files."
   MsgBox Msg  ' Display message.
   Kill "TESTFILE.*" ' Remove test files.
   File1.Refresh  ' Update file list box.
End Sub
```

SetFocus Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthSetFocusC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthSetFocusX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthSetFocusA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthSetFocusS"}

Moves the focus to the specified control or form.

Syntax

object.**SetFocus**

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

Remarks

The object must be a **Form** object, **MDIForm** object, or control that can receive the focus. After invoking the **SetFocus** method, any user input is directed to the specified form or control.

You can only move the focus to a visible form or control. Because a form and controls on a form aren't visible until the form's Load event has finished, you can't use the **SetFocus** method to move the focus to the form being loaded in its own Load event unless you first use the **Show** method to show the form before the Form_Load event procedure is finished.

You also can't move the focus to a form or control if the **Enabled** property is set to **False**. If the **Enabled** property has been set to **False** at design time, you must first set it to **True** before it can receive the focus using the **SetFocus** method.

DblClick Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"formDblClickSee;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtDblClickX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"formDblClickA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtDblClickS"}

Occurs when the user presses and releases a mouse button and then presses and releases it again over an object.

For a form, the DblClick event occurs when the user double-clicks a disabled control or a blank area of a form. For a control, it occurs when the user:

- Double-clicks a control with the left mouse button.
- Double-clicks an item in a **ComboBox** control whose **Style** property is set to 1 (Simple) or in a **FileListBox**, **ListBox**, **DBCombo**, or **DBList** control.

Syntax

Private Sub Form_DblClick ( )
Private Sub *object*_**DblClick (***index* **As Integer)**

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	Identifies the control if it's in a control array.

Remarks

The argument *Index* uniquely identifies a control if it's in a control array. You can use a DblClick event procedure for an implied action, such as double-clicking an icon to open a window or document. You can also use this type of procedure to carry out multiple steps with a single action, such as double-clicking to select an item in a list box and to close the dialog box.

To produce such shortcut effects in Visual Basic, you can use a DblClick event procedure for a list box or file list box in tandem with a default button — a **CommandButton** control with its **Default** property set to **True**. As part of the DblClick event procedure for the list box, you simply call the default button's Click event.

For those objects that receive Mouse events, the events occur in this order: MouseDown, MouseUp, Click, DblClick, and MouseUp.

If DblClick doesn't occur within the system's double-click time limit, the object recognizes another Click event. The double-click time limit may vary because the user can set the double-click speed in the Control Panel. When you're attaching procedures for these related events, be sure that their actions don't conflict. Controls that don't receive DblClick events may receive two clicks instead of a DblClick.

Note To distinguish between the left, right, and middle mouse buttons, use the MouseDown and MouseUp events.

If there is code in the Click event, the DlbClick event will never trigger.

DblClick Event Example

This example displays a selected list item in a **TextBox** control when either a **CommandButton** control is clicked or a list item is double-clicked. To try this example, paste the code into the Declarations section of a **Form** object that contains a **ListBox** control, a **TextBox** control, and a **CommandButton** control. Then run the example and click the **CommandButton** control or double-click an item in the **ListBox** control.

```
Private Sub Form_Load ()
   List1.AddItem "John" ' Add list box entries.
   List1.AddItem "Paul"
   List1.AddItem "George"
   List1.AddItem "Ringo"
End Sub

Private Sub List1_DblClick ()
   Command1.Value = True   ' Trigger Click event.
End Sub

Private Sub Command1_Click ()
   Text1.Text = List1.Text ' Display selection.
End Sub
```

HelpFile Property (App, CommonDialog, MenuLine)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHelpFileC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHelpFileX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHelpFileA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHelpFileS"}

Specifies the path and filename of a Microsoft Windows Help file used by your application to display Help or online documentation.

Syntax

object.**HelpFile**[= *filename*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
filename	A string expression specifying the path and filename of the Windows Help file for your application.

Remarks

If you've created a Windows Help file for your application and set the application's **HelpFile** property, Visual Basic automatically calls Help when a user presses the F1 key. If there is a context number in the **HelpContextID** property for either the active control or the active form, Help displays a topic corresponding to the current Help context; otherwise it displays the main contents screen.

You can also use the **HelpFile** property to determine which Help file is displayed when a user requests Help from the Object Browser for an ActiveX component.

Note Building a Help file requires the Microsoft Windows Help Compiler, which is available with Visual Basic, Professional Edition.

HelpFile Property Example

This example uses topics in the Visual Basic Help file and demonstrates how to specify context numbers for Help topics. To try this example, paste the code into the Declarations section of a **Form** object that contains a **TextBox** control and a **Frame** control with an **OptionButton** control inside of it. Run the example. Once the program is running, move the focus to one of the components, and press F1.

```
' Actual context numbers from the Visual Basic Help file.
' Define constants.
Const winPictureBox = 2016002
Const winCommandButton = 2007557

Private Sub Form_Load ()
   App.HelpFile = "VB5.HLP"
   Text1.HelpContextID = winPictureBox
   Form1.HelpContextID = winCommandButton
End Sub
```

Clipboard Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjClipboardC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjClipboardX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjClipboardP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjClipboardM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjClipboardE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjClipboardS"}

Provides access to the system Clipboard.

Syntax

Clipboard

Remarks

The **Clipboard** object is used to manipulate text and graphics on the Clipboard. You can use this object to enable a user to copy, cut, and paste text or graphics in your application. Before copying any material to the **Clipboard** object, you should clear its contents by as performing a **Clear** method, such as `Clipboard.Clear`.

Note that the **Clipboard** object is shared by all Windows applications, and thus, the contents are subject to change whenever you switch to another application.

The **Clipboard** object can contain several pieces of data as long as each piece is in a different format. For example, you can use the **SetData** method to put a bitmap on the **Clipboard** with the **vbCFDIB** format, and then use the **SetText** method with the **vbCFText** format to put text on the **Clipboard**. You can then use the **GetText** method to retrieve the text or the **GetData** method to retrieve the graphic. Data on the **Clipboard** is lost when another set of data of the same format is placed on the **Clipboard** either through code or a menu command.

Printer Object, Printers Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjPrinterC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjPrinterX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjPrinterP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjPrinterM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjPrinterE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjPrinterS"}

The **Printer** object enables you to communicate with a system printer (initially the default system printer).

The **Printers** collection enables you to gather information about all the available printers on the system.

Syntax

Printer
Printers(*index***)**

The *index* placeholder represents an integer with a range from 0 to `Printers.Count-1`.

Remarks

Use graphics methods to draw text and graphics on the **Printer** object. Once the **Printer** object contains the output you want to print, you can use the **EndDoc** method to send the output directly to the default printer for the application.

You should check and possibly revise the layout of your forms if you print them. If you use the **PrintForm** method to print a form, for example, graphical images may be clipped at the bottom of the page and text carried over to the next page.

The **Printers** collection enables you to query the available printers so you can specify a default printer for your application. For example, you may want to find out which of the available printers uses a specific printer driver. The following code searches all available printers to locate the first printer with its page orientation set to portrait, then sets it as the default printer:

```
Dim X As Printer
For Each X In Printers
   If X.Orientation = vbPRORPortrait Then
      ' Set printer as system default.
      Set Printer = X
      ' Stop looking for a printer.
      Exit For
   End If
Next
```

You designate one of the printers in the **Printers** collection as the default printer by using the **Set** statement. The preceding example designates the printer identified by the object variable `X`, the default printer for the application.

Note If you use the **Printers** collection to specify a particular printer, as in `Printers(3)`, you can only access properties on a read-only basis. To both read and write the properties of an individual printer, you must first make that printer the default printer for the application.

Font Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjFontC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjFontX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjFontP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjFontM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjFontE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjFontS"}

The **Font** object contains information needed to format text for display in the interface of an application or for printed output.

Syntax

Font

Remarks

You frequently identify a **Font** object using the **Font** property of an object that displays text (such as a **Form** object or the **Printer** object).

You cannot create a **Font** object using code like `Dim X As New Font`. If you want to create a **Font** object, you must use the **StdFont** object like this:

```
Dim X As New StdFont
```

If you put a **TextBox** control named Text1 on a form, you can dynamically change it's font **Font** object to another using the **Set** statement, as in the following example:

```
Dim X As New StdFont
X.Bold = True
X.Name = "Arial"
Set Text1.Font = X
```

Load Statement

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbstmLoadC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbstmLoadX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbstmLoadS"}

Loads a form or control into memory.

Syntax

Load*object*

The *object* placeholder is the name of a **Form** object, **MDIForm** object, or control array element to load.

Remarks

You don't need to use the **Load** statement with forms unless you want to load a form without displaying it. Any reference to a form (except in a **Set** or **If...TypeOf** statement) automatically loads it if it's not already loaded. For example, the **Show** method loads a form before displaying it. Once the form is loaded, its properties and controls can be altered by the application, whether or not the form is actually visible. Under some circumstances, you may want to load all your forms during initialization and display them later as they're needed.

When Visual Basic loads a **Form** object, it sets form properties to their initial values and then performs the Load event procedure. When an application starts, Visual Basic automatically loads and displays the application's startup form.

If you load a **Form** whose **MDIChild** property is set to **True** (in other words, the child form) before loading an **MDIForm**, the **MDIForm** is automatically loaded before the child form. MDI child forms cannot be hidden, and thus are immediately visible after the Form_Load event procedure ends.

The standard dialog boxes produced by Visual Basic functions such as **MsgBox** and **InputBox** do not need to be loaded, shown, or unloaded, but can simply be invoked directly.

Load Statement Example

This example uses the **Load** statement to load a **Form** object. To try this example, paste the code into the Declarations section of a **Form** object, and then run the example and click the **Form** object.

```
Private Sub Form_Click ()
   Dim Answer, Msg as String ' Declare variable.
   Unload Form1   ' Unload form.
   Msg = "Form1 has been unloaded. Choose Yes to load and "
   Msg = Msg & "display the form. Choose No to load the form "
   Msg = Msg & "and leave it invisible."
   Answer = MsgBox(Msg, vbYesNo)   ' Get user response.
   If Answer = vbYes Then  ' Evaluate answer.
      Show  ' If Yes, show form.
   Else
      Load Form1  ' If No, just load it.
      Msg = "Form1 is now loaded. Choose OK to display it."
      MsgBox Msg  ' Display message.
      Show  ' Show form.
   End If
End Sub
```

Unload Statement

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbstmUnloadC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vastmUnloadX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbstmUnloadS"}

Unloads a form or control from memory.

Syntax

Unload *object*

The *object* placeholder is the name of a **Form** object or control array element to unload.

Remarks

Unloading a form or control may be necessary or expedient in some cases where the memory used is needed for something else, or when you need to reset properties to their original values.

Before a form is unloaded, the Query_Unload event procedure occurs, followed by the Form_Unload event procedure. Setting the *cancel* argument to **True** in either of these events prevents the form from being unloaded. For **MDIForm** objects, the **MDIForm** object's Query_Unload event procedure occurs, followed by the Query_Unload event procedure and Form_Unload event procedure for each MDI child form, and finally the **MDIForm** object's Form_Unload event procedure.

When a form is unloaded, all controls placed on the form at run time are no longer accessible. Controls placed on the form at design time remain intact; however, any run-time changes to those controls and their properties are lost when the form is reloaded. All changes to form properties are also lost. Accessing any controls on the form causes it to be reloaded.

Note When a form is unloaded, only the displayed component is unloaded. The code associated with the form module remains in memory.

Only control array elements added to a form at run time can be unloaded with the **Unload** statement. The properties of unloaded controls are reinitialized when the controls are reloaded.

Unload Statement Example

This example uses the **Unload** statement to unload a **Form** object. To try this example, paste the code into the Declarations section of a **Form** object, and then run the example and click the **Form** object.

```
Private Sub Form_Click ()
   Dim Answer, Msg   ' Declare variable.
   Unload Form1   ' Unload form.
   Msg = "Form1 has been unloaded. Choose Yes to load and "
   Msg = Msg & "display the form. Choose No to load the form "
   Msg = Msg & "and leave it invisible."
   Answer = MsgBox(Msg, vbYesNo)   ' Get user response.
   If Answer = vbYes Then  ' Evaluate answer.
      Show  ' If Yes, show form.
   Else
      Load Form1  ' If No, just load it.
      Msg = "Form1 is now loaded. Choose OK to display it."
      MsgBox Msg  ' Display message.
      Show  ' Show form.
   End If
End Sub
```

LoadPicture Function

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbfctLoadPictureC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbfctLoadPictureX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbfctLoadPictureS"}

Loads a graphic into a form's **Picture** property, a **PictureBox** control, or **Image** control.

Syntax

LoadPicture([*stringexpression*]**)**

The *stringexpression* argument is the name of a graphics file to be loaded.

Remarks

Graphics formats recognized by Visual Basic include bitmap (.bmp) files, icon (.ico) files, run-length encoded (.rle) files, metafile (.wmf) files, enhanced metafiles (.emf), GIF files, and JPEG (.jpg) files.

Graphics are cleared from forms, picture boxes, and image controls by assigning **LoadPicture** with no argument.

To load graphics for display in a **PictureBox** control, **Image** control, or as the background of a form, the return value of **LoadPicture** must be assigned to the **Picture** property of the object on which the picture is displayed. For example:

```
Set Picture = LoadPicture("PARTY.BMP")
Set Picture1.Picture = LoadPicture("PARTY.BMP")
```

To assign an icon to a form, set the return value of the **LoadPicture** function to the **Icon** property of the **Form** object:

```
Set Form1.Icon = LoadPicture("MYICON.ICO")
```

Icons can also be assigned to the **DragIcon** property of all controls except **Timer** controls and **Menu** controls. For example:

```
Set Command1.DragIcon = LoadPicture("MYICON.ICO")
```

Load a graphics file into the system Clipboard using **LoadPicture** as follows:

```
Clipboard.SetData LoadPicture("PARTY.BMP")
```

LoadPicture Function Example

This example uses the **LoadPicture** function to load a picture into a form's **Picture** property and to clear the picture from the **Form** object. To try this example, paste the code into the Declarations section of a **Form** object, and then run the example and click the **Form** object.

```
Private Sub Form_Click ()
   Dim Msg as String ' Declare variables.
   On Error Resume Next ' Set up error handling.
   Height = 3990
   Width = 4890   ' Set height and width.
   Set Picture = LoadPicture("PAPER.BMP")
   ' Load bitmap.
   If Err Then
      Msg = "Couldn't find the .BMP file."
      MsgBox Msg  ' Display error message.
      Exit Sub ' Quit if error occurs.
   End If
   Msg = "Choose OK to clear the bitmap from the form."
   MsgBox Msg
   Set Picture = LoadPicture()     ' Clear form.
End Sub
```

SavePicture Statement

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbstmSavePictureC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbstmSavePictureX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbstmSavePictureS"}

Saves a graphic from the **Picture** or **Image** property of a **Form** object, or the **Picture** property of a **PictureBox** control or **Image** control, to a file.

Syntax

SavePicture *picture*, *stringexpression*

The **SavePicture** statement syntax has these parts:

Part	Description
picture	**Picture** or **Image** control from which the graphics file is to be created.
stringexpression	Filename of the graphics file to save.

Remarks

If a graphic was loaded from a file to the **Picture** property of an object, either at design time or at run time, and it's a bitmap, icon, metafile, or enhanced metafile, it's saved using the same format as the original file. If it is a GIF or JPEG file, it is saved as a bitmap file.

Graphics in an **Image** property are always saved as bitmap (.bmp) files regardless of their original format.

SavePicture Statement Example

This example uses the **SavePicture** statement to save a graphic drawn into a **Form** object's **Picture** property. To try this example, paste the code into the Declarations section of a **Form** object, and then run the example and click the **Form** object.

```
Private Sub Form_Click ()
   ' Declare variables.
   Dim CX, CY, Limit, Radius as Integer, Msg as String
   ScaleMode = vbPixels ' Set scale to pixels.
   AutoRedraw = True ' Turn on AutoRedraw.
   Width = Height ' Change width to match height.
   CX = ScaleWidth / 2  ' Set X position.
   CY = ScaleHeight / 2 ' Set Y position.
   Limit = CX  ' Limit size of circles.
   For Radius = 0 To Limit ' Set radius.
      Circle (CX, CY), Radius, RGB(Rnd * 255, Rnd * 255, Rnd * 255)
      DoEvents ' Yield for other processing.
   Next Radius
   Msg = "Choose OK to save the graphics from this form "
   Msg = Msg & "to a bitmap file."
   MsgBox Msg
   SavePicture Image, "TEST.BMP"   ' Save picture to file.
End Sub
```

Circle Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthCircleC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthCircleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthCircleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthCircleS"}

Draws a circle, ellipse, or arc on an object.

Syntax

object.**Circle Step (***x, y***),** *radius,* [*color, start, end, aspect*]

The **Circle** method syntax has the following object qualifier and parts.

Part	Description
object	Optional. Object expression that evaluates to an object in the Applies To list. If *object* is omitted, the **Form** with the focus is assumed to be *object*.
Step	Optional. Keyword specifying that the center of the circle, ellipse, or arc is relative to the current coordinates given by the **CurrentX** and **CurrentY** properties of *object*.
(*x*, *y***)**	Required. **Single** values indicating the coordinates for the center point of the circle, ellipse, or arc. The **ScaleMode** property of *object* determines the units of measure used.
radius	Required. **Single** value indicating the radius of the circle, ellipse, or arc. The **ScaleMode** property of *object* determines the unit of measure used.
color	Optional. **Long** integer value indicating the RGB color of the circle's outline. If omitted, the value of the **ForeColor** property is used. You can use the **RGB** function or **QBColor** function to specify the color.
start, *end*	Optional. Single-precision values. When an arc or a partial circle or ellipse is drawn, *start* and *end* specify (in radians) the beginning and end positions of the arc. The range for both is -2 pi radians to 2 pi radians. The default value for *start* is 0 radians; the default for *end* is 2 * pi radians.
aspect	Optional. Single-precision value indicating the aspect ratio of the circle. The default value is 1.0, which yields a perfect (non-elliptical) circle on any screen.

Remarks

To fill a circle, set the **FillColor** and **FillStyle** properties of the object on which the circle or ellipse is drawn. Only a closed figure can be filled. Closed figures include circles, ellipses, or pie slices (arcs with radius lines drawn at both ends).

When drawing a partial circle or ellipse, if *start* is negative, **Circle** draws a radius to *start*, and treats the angle as positive; if *end* is negative, **Circle** draws a radius to *end* and treats the angle as positive. The **Circle** method always draws in a counter-clockwise (positive) direction.

The width of the line used to draw the circle, ellipse, or arc depends on the setting of the **DrawWidth** property. The way the circle is drawn on the background depends on the setting of the **DrawMode** and **DrawStyle** properties.

When drawing pie slices, to draw a radius to angle 0 (giving a horizontal line segment to the right), specify a very small negative value for *start*, rather than zero.

You can omit an argument in the middle of the syntax, but you must include the argument's comma before including the next argument. If you omit an optional argument, omit the comma following the last argument you specify.

When **Circle** executes, the **CurrentX** and **CurrentY** properties are set to the center point specified by the arguments.

Circle Method Example

This example uses the **Circle** method to draw a number of concentric circles in the center of a form. To try this example, paste the code into the General section of a form. Then press F5 and click the form.

```
Sub Form_Click ()
   Dim CX, CY, Radius, Limit ' Declare variable.
   ScaleMode = 3  ' Set scale to pixels.
   CX = ScaleWidth / 2  ' Set X position.
   CY = ScaleHeight / 2 ' Set Y position.
   If CX > CY Then Limit = CY Else Limit = CX
   For Radius = 0 To Limit ' Set radius.
      Circle (CX, CY), Radius,RGB(Rnd * 255, Rnd * 255, Rnd * 255)
   Next Radius
End Sub
```

Line Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthLineC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthLineX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthLineA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthLineS"}

Draws lines and rectangles on an object.

Syntax

object.**Line Step (***x1***,** *1***) Step (***x2***,** *y2***),** *color***, BF**

The **Line** method syntax has the following object qualifier and parts:

Part	Description
object	Optional. Object expression that evaluates to an object in the Applies To list. If object is omitted, the **Form** with the focus is assumed to be *object*.
Step	Optional. Keyword specifying that the starting point coordinates are relative to the current graphics position given by the **CurrentX** and **CurrentY** properties.
(*x1*, *y1***)**	Optional. **Single** values indicating the coordinates of the starting point for the line or rectangle. The **ScaleMode** property determines the unit of measure used. If omitted, the line begins at the position indicated by **CurrentX** and **CurrentY**.
Step	Optional. Keyword specifying that the end point coordinates are relative to the line starting point.
(*x2*, *y2***)**	Required. **Single** values indicating the coordinates of the end point for the line being drawn.
color	Optional. **Long** integer value indicating the RGB color used to draw the line. If omitted, the **ForeColor** property setting is used. You can use the **RGB** function or **QBColor** function to specify the color.
B	Optional. If included, causes a box to be drawn using the coordinates to specify opposite corners of the box.
F	Optional. If the **B** option is used, the **F** option specifies that the box is filled with the same color used to draw the box. You cannot use **F** without **B**. If **B** is used without **F**, the box is filled with the current **FillColor** and **FillStyle**. The default value for **FillStyle** is transparent.

Remarks

To draw connected lines, begin a subsequent line at the end point of the previous line.

The width of the line drawn depends on the setting of the **DrawWidth** property. The way a line or box is drawn on the background depends on the setting of the **DrawMode** and **DrawStyle** properties.

When **Line** executes, the **CurrentX** and **CurrentY** properties are set to the end point specified by the arguments.

Line Method Example

This example uses the **Line** method to draw concentric boxes on a form. To try this example, paste the code into the General section of a form. Then press F5 and click the form.

```
Sub Form_Click ()
   Dim CX, CY, F, F1, F2, I  ' Declare variables
   ScaleMode = 3  ' Set ScaleMode to pixels.
   CX = ScaleWidth / 2  ' Get horizontal center.
   CY = ScaleHeight / 2 ' Get vertical center.
   DrawWidth = 8  ' Set DrawWidth.
   For I = 50 To 0 Step -2
      F = I / 50  ' Perform interim
      F1 = 1 - F: F2 = 1 + F ' calculations.
      Forecolor = QBColor(I Mod 15)      ' Set foreground color.
      Line (CX * F1, CY * F1)-(CX * F2, CY * F2), , BF
   Next I
   DoEvents ' Yield for other processing.
   If CY > CX Then   ' Set DrawWidth.
      DrawWidth = ScaleWidth / 25
   Else
      DrawWidth = ScaleHeight / 25
   End If
   For I = 0 To 50 Step 2  ' Set up loop.
      F = I / 50  ' Perform interim
      F1 = 1 - F: F2 = 1 + F ' calculations.
      Line (CX * F1, CY)-(CX, CY * F1)   ' Draw upper-left.
      Line -(CX * F2, CY)  ' Draw upper-right.
      Line -(CX, CY * F2)  ' Draw lower-right.
      Line -(CX * F1, CY)  ' Draw lower-left.
      Forecolor = QBColor(I Mod 15)      ' Change color each time.
   Next I
   DoEvents ' Yield for other processing.
End Sub
```

PSet Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthPSetC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthPSetX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthPSetA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthPSetS"}

Sets a point on an object to a specified color.

Syntax

object.**PSet Step (***x***,** *y***),** *color*

The **PSet** method syntax has the following object qualifier and parts:

Part	Description
object	Optional. Object expression that evaluates to an object in the Applies To list. If *object* is omitted, the **Form** with the focus is assumed to be *object*.
Step	Optional. Keyword specifying that the coordinates are relative to the current graphics position given by the **CurrentX** and **CurrentY** properties.
(*x*, *y***)**	Required. **Single** values indicating the horizontal (x-axis) and vertical (y-axis) coordinates of the point to set.
color	Optional. **Long** integer value indicating the RGB color specified for point. If omitted, the current **ForeColor** property setting is used. You can use the **RGB** function or **QBColor** function to specify the color.

Remarks

The size of the point drawn depends on the setting of the **DrawWidth** property. When **DrawWidth** is 1, **PSet** sets a single pixel to the specified color. When **DrawWidth** is greater than 1, the point is centered on the specified coordinates.

The way the point is drawn depends on the setting of the **DrawMode** and **DrawStyle** properties.

When **PSet** executes, the **CurrentX** and **CurrentY** properties are set to the point specified by the arguments.

To clear a single pixel with the **PSet** method, specify the coordinates of the pixel and use the **BackColor** property setting as the *color* argument.

PSet Method Example

This example uses the **PSet** method to draw confetti on a form. To try this example, paste the code into the General section of a form. Then press F5 and click the form.

```
Sub Form_Click ()
   Dim CX, CY, Msg, XPos, YPos     ' Declare variables.
   ScaleMode = 3  ' Set ScaleMode to
         ' pixels.
   DrawWidth = 5  ' Set DrawWidth.
   ForeColor = QBColor(4)  ' Set foreground to red.
   FontSize = 24  ' Set point size.
   CX = ScaleWidth / 2  ' Get horizontal center.
   CY = ScaleHeight / 2 ' Get vertical center.
   Cls   ' Clear form.
   Msg = "Happy New Year!"
   CurrentX = CX - TextWidth(Msg) / 2    ' Horizontal position.
   CurrentY = CY - TextHeight(Msg) ' Vertical position.
   Print Msg   ' Print message.
   Do
      XPos = Rnd * ScaleWidth      ' Get horizontal position.
      YPos = Rnd * ScaleHeight     ' Get vertical position.
      PSet (XPos, YPos), QBColor(Rnd * 15)     ' Draw confetti.
      DoEvents ' Yield to other
   Loop  ' processing.
End Sub
```

You need to fill in the RecordSource property of the Data control to get a list of field names.

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

Before you can see a list of possible settings for the **DataField** property of a bound control, you must first set the **RecordSource** property of the **Data** control.

You need to fill in the DatabaseName and/or Connect properties of the Data Control to complete this operation

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

Before you can see a list of possible settings for the **RecordSource** property of the **Data** control, you must first set at least the **DatabaseName** property (for Access database .MDB files), and possibly the **Connect** property (for all non-Access database files).

Line 'item1': 'item2' has a quoted string where the property name should be.

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

A quoted string appeared in the ASCII form file where the property name was expected. Property names are not placed inside quotation marks.

Line 'item1': All controls must precede menus; cannot load control 'item2'.

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

A control appeared in an incorrect location in the ASCII form file. All controls must be loaded before menus.

Line 'item1': Cannot load control 'item2'; containing control not a valid container.

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

You attempted to load a control into a control which isn't a valid container.

Line 'item1': Cannot load control 'item2'; name already in use.

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

The control named in the ASCII text file couldn't be loaded because its name is already in use elsewhere in the application.

Line 'item1': Cannot set checked property in menu 'item2'. Parent menu cannot be checked.

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

A top-level **Menu** control appeared in the ASCII form file with its **Checked** property set to **True**. Top-level menus can't be checked. The **Menu** control will be loaded, but its **Checked** property won't be set.

Line 'item1': Cannot set Shortcut property in menu 'item2'. Parent menu cannot have a shortcut key.

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

A top-level **Menu** control appeared in the ASCII form file with a shortcut key defined. Top-level menus can't have a shortcut key. The **Menu** control will be loaded, but its **Shortcut** property won't be set.

Line 'item1': Class 'item2' of control 'item3' was not a loaded control class.

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

The ASCII file contains a control class that Visual Basic doesn't recognize. Add the custom control with this class to your project.

Line 'item1': Class name too long; truncated to 'item2'.

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

The ASCII file contains a class name longer than 40 characters. The class will be loaded with the name truncated to 40 characters.

Line 'item1': Control name too long; truncated to 'item2'.

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

The ASCII file contains a control name longer than 40 characters. The control will be loaded with the name truncated to 40 characters.

Line 'item1': Did not find an index property and control 'item2' already exists.

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

The control can't be loaded because there is no index, and it has the same name as a previously loaded control.

Line 'item1': Maximum nesting level for controls exceeded with 'item2'.

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

The ASCII file contains controls nested more than seven levels deep.

Line 'item1': Missing or invalid control class in file 'item2'.

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

The ASCII file contains an unknown control class in the form description, or the class name isn't a valid string in Visual Basic.

Line 'item1': Missing or invalid control name in file 'item2'.

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

The ASCII file contains an unknown control name in the form description, or the control name isn't a valid string in Visual Basic.

Line 'item1': Parent menu 'item2' cannot be loaded as a separator.

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

The ASCII file contains a **Menu** control whose parent or top-level menu is defined as a menu separator. Top-level menus can't be menu separators. The separator won't be set.

Line 'item1': Property 'item2' in 'item3' could not be loaded.

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

The ASCII file contains an unknown property. The property will be skipped when loading the form.

Line 'item1': Property 'item2' in 'item3' could not be set.

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

Visual Basic can't set the property of the specified control as indicated by the form description in the ASCII file.

Line 'item1': Property 'item2' in 'item3' had an invalid file reference.

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

The ASCII file contains a reference to a file that Visual Basic couldn't find in the specified directory.

Line 'item1': Property 'item2' in 'item3' had an invalid value.

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

The ASCII file contains a property with a value that isn't correct for this control. The property is set with its default value.

Line 'item1': Property 'item2' in 'item3' must be a quoted string.

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

The ASCII file contains a property that should appear inside quotation marks, but the quotation marks are missing. This line in the form description is ignored.

Line 'item1': Property 'item2' in control 'item3' had an invalid property index.

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

The ASCII file contains a property name with a property index greater than 255.

Line 'item1': Syntax error: property 'item2' in 'item3' was missing an '='.

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

The ASCII file contains a property name and value without an equal sign between them. The property isn't loaded.

Line 'item1': The control name 'item2' is invalid.

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

The ASCII file contains a control name that isn't a valid string in Visual Basic. The control isn't loaded.

Line 'item1': The Form or MDIForm name 'item2' is already in use; cannot load this form.

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

The ASCII file contains a form with a name that is already being used elsewhere in the application. The form isn't loaded.

Line 'item1': The Form or MDIForm name 'item2' is not valid; cannot load this form.

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

The ASCII file contains a form name that isn't a valid string in Visual Basic. The form isn't loaded.

Line 'item1': The property name 'item2' in 'item3' is invalid.

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

The ASCII file contains a property name that isn't a valid property for that control.

A file must be specified in order to create a compatible ActiveX component.

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

You must specify a file when you create a version compatible ActiveX component. On the Components tab of the Project Properties window, type in the name of an ActiveX component in the text box, or click the ellipse button to browse for one.

A procedure of that name already exists

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

You tried to add a procedure but that procedure name already exists. This error has the following cause and solution:

- In the **Add Procedure** dialog box, you tried to add a procedure that already exists in the project. Change the name of your procedure to avoid this problem.

All 'item1' objects in this project will be upgraded to 'item2' objects

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgAllitem1ObjectsInThisProjectWillBeUpgradedToitem2ObjectsC"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgAllitem1ObjectsInThisProjectWillBeUpgradedToitem2ObjectsS"}

You're trying to load a project that contains an OLE Embedding (Excel, Word, etc.) and the server application on your machine is a more recent version than the one used to save the project. Your project should continue to load correctly.

An error occurred while background loading module 'item'. Background load will now abort and the code for some modules may not be loaded. Saving these modules to their current file name will result in code loss. Please load a new project.

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

An error occurred while background loading your project. This error has the following causes and solutions:

- While the source file was loading, it may have been changed or corrupted by an editor other than Visual Basic.

  Do not modify source code from a separate editor while it is being loaded into Visual Basic.
- During the load of your project or projects, the drive containing the source files became unavailable.

  If your source files are on a network, verify that the network is still connected.

If this error persists, turn off the Background Project Load option from the Advanced tab of the **Options** dialog box.

An instance of 'item' cannot be created because it's designer window is open

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

You cannot create an instance of a form when its designer window is open.

Occurs if code running at design time tries to create an instance of a form (such as Form1.Show), and its designer is open.

Can't find file 'item'

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

The specified file was not found. This error has the following cause and solution:

- You tried to load a project but the specified file doesn't exist.

  You may have moved or deleted a file belonging to this project. Verify that the file is available on your drive.

Can't run without setting a Startup Project first

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

You must set a startup project before running. This error has the following causes and solutions:

- You had a startup project and two or more other projects in the project group that could be a startup project, but you removed the one that was a startup project.

  Make one of the projects a startup project by right clicking the project in the Project Explorer and selecting the Set as **Start Up** menu item.
- This error also occurs in any other case where there is more than one project that could be a startup project after removing the current startup project.

  Make one of the projects a startup project by right clicking the project in the Project Explorer and selecting the Set as **Start Up** menu item.

Can't set the project name at this time

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

The project name cannot be set.

Compile Error in File 'item1' : 'item2'

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgCompileErrorInFileitem1item2C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgCompileErrorInFileitem1item2S"}

There was an error compiling the specified source file during a command line build. Check the file and correct the error.

Compile Error in File 'item1', Line 'item2' : 'item3'

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgCompileErrorInFileitem1Lineitem2item3C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgCompileErrorInFileitem1Lineitem2item3S"}

There was an error compiling the source file at the specified line number during a command line build. Check the file and correct the error.

Conflicting attributes were found in 'item'. The defaults will be used

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

Some of the attribute statements in a Form, User Control, Property Page, User Document, or Class Module conflict with the required settings for their type. For example, forms must always have the **VB_PredeclaredId** attribute equal to **True**.

This error can occur if the file was modified by an editor other than Visual Basic.

Conflicting names were found in 'item1'. The name 'item2' will be used

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgConflictingNamesWereFoundInitem1TheNameitem2WillBeUsedC"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgConflictingNamesWereFoundInitem1TheNameitem2WillBeUsedS"}

The name of a form occurs twice in the source file for a Form, User Control, Property Page, or User Document. The names are different ,so the specified name will be used. This error can occur if the file was modified by an editor other than Visual Basic.

Could not create reference: 'item'

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

There was an error establishing a reference while loading the file, so the reference was not added.

Display more load errors?

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

There were errors loading your .VBP project file. Select **Yes** to display more load errors. Choose **No** to close.

Errors occurred during load

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

There were problems loading a form at design time.

This error can occur when there is an invalid property value in an .FRM file, however, Visual Basic will continue loading the form.

File is read-only

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgFileIsReadonlyvb5C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgFileIsReadonlyvb5S"}

The file is read-only and cannot be changed. This error has the following cause and solution:

- You are trying to edit a form whose file attribute is set to read-only. You cannot modify form files that are read-only.

  You must change the file attribute to read-write in order to modify the form.

Invalid Base Address

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

You entered an invalid base address. This error has the following cause and solution:

- On the Compile Tab in the **Project Properties** dialog, you specified an invalid Base Address. A valid base address must be a multiple of 64K, greater than 64K, and less than &H7FFF0000.

This error occurs in ActiveX DLL or ActiveX Control projects.

Invalid number of threads. The number of threads should be integer between 1 and 32767

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgInvalidNumberOfThreadsTheNumberOfThreadsShouldBeIntegerBetween132767C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgInvalidNumberOfThreadsTheNumberOfThreadsShouldBeIntegerBetween132767S"}

You entered an invalid number of threads for the Thread Pool in the General Tab of the **Project Properties** dialog box. The number of threads must be an integer between 1 and 32767.

'item' can not be public in this type of project. The item has been changed to private

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

The item cannot be public in this type of project and will be changed to private.

Visual Basic changes some items to private in order to avoid inconsistencies in the project. For example, if Visual Basic detects that the project has a public class module but the Project Type is set to a type that doesn't support public class modules, the public class module will be changed to private.

This error can occur if any of the files in the project have been modified in an editor other than Visual Basic. For example, Visual Basic detects that Standard Project Executable owns public class modules while this type of project allows only private class modules.

'item' can not be public in this type of project. Use Project Properties to change the project type

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

This occurs when you try to change a User Control's **Public** property to **True** when the User Control is not in an ActiveX control project. Either change the Project Type on the General Tab of the **Project Properties** dialog to ActiveX control, or don't set the User Control's **Public** property to **True**.

'item' could not be loaded

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

The specified file could not be loaded. The file may be corrupt or you may not have rights access to this file.

'item1' is an invalid key. The file 'item2' can't be loaded

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgitem1IsInvalidKeyTheFileitem2CantBeLoadedC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgitem1IsInvalidKeyTheFileitem2CantBeLoadedS"}

Visual Basic did not recognize the specified key in the .VBP file. This error has the following cause and solution:

- The .VBP file contains a key that Visual Basic doesn't recognize. The file may be corrupt.
  Install a backup copy of the specified file or modify the file using a text editor to correct the error.

Warning Modifying a Visual Basic file in an editor other than Visual Basic may corrupt the file. This is only recommended for advanced users.

'item1' is referenced by 'item2' project and cannot be updated.

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgitem1IsReferencedByitem2ProjectCannotBeUpdatedC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgitem1IsReferencedByitem2ProjectCannotBeUpdatedS"}

You are trying to modify the specified control but it is shared between two or more projects. This error has the following cause and solution:

- You have two or more projects in the same project group that use the same control and are trying to modify the control.

  You cannot update a control that is referenced by two or more projects.

'item' will not be loaded. Name is already in use

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

Form names must be unique for each project. This error has the following cause and solution:

- You tried to load a form with the same name as a form that is already in the project.
  To avoid this error, change the **Name** property of the duplicate form to another name.

Line 'item1': Cannot create embedded object in 'item2'

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgLineitem1CannotCreateEmbeddedObjectInitem2C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgLineitem1CannotCreateEmbeddedObjectInitem2S"}

An embedded object could not be created during the load of a Form, User Control, User Document, or Property Page from a text file. For example, if you had previously inserted a Microsoft Word Document onto the form, then removed Microsoft Word from your system you would get this error. This message is written to the error log file.

Line 'item1': Cannot create embedded object in 'item2'; license not found

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgLineitem1CannotCreateEmbeddedObjectInitem2LicenseNotFoundC"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgLineitem1CannotCreateEmbeddedObjectInitem2LicenseNotFoundS"}

An embedded object could not be created during the load of a Form, User Control, User Document, or Property Page from a text file, due to the license file not being found. You must have a license to use this object. Check with the object's vendor for more information. This message is written to the error log file.

Line 'item1': Cannot load control 'item2'

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgLineitem1CannotLoadControlitem2C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgLineitem1CannotLoadControlitem2S"}

An ActiveX control could not be created during the load of a Form, User Control, User Document, or Property Page from a text file. The control may be missing or corrupted. Reinstall the control and try again or check with the control's vendor for more information. This message is written to the error log file.

Line 'item1': Cannot load control 'item2'; license not found

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgLineitem1CannotLoadControlitem2LicenseNotFoundC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgLineitem1CannotLoadControlitem2LicenseNotFoundS"}

An ActiveX control could not be created during the load of a Form, User Control, User Document, or Property Page from a text file, due to the license file not being found. You must have a license to use this control. Check with the control's vendor for more information. This message is written to the error log file.

Line 'item1': Could not create reference: 'item2'

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgLineitem1CouldNotCreateReferenceitem2C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgLineitem1CouldNotCreateReferenceitem2S"}

The specified reference could not be created.

Line 'item1': The CLSID 'item2' for 'item3' is invalid

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgLineitem1TheCLSIDitem2Foritem3IsInvalidC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgLineitem1TheCLSIDitem2Foritem3IsInvalidS"}

An object could not be loaded during the load of a Form, User Control, User Document, or Property Page from a text file. The CLSID specified in the file is not valid. Applies only to objects that are properties, such as the **Font** object. This message is written to the error log file.

MDI/SDI option change will take effect the next time the Development Environment is started

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

The SDI Development Environment option on the Advanced tab in the **Options** dialog box doesn't take effect until you restart Visual Basic.

MultiSelect must be 0 - None when Style is 1 - Checkbox

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgMultiSelectMustBe0NoneWhenStyleIs1CheckboxC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgMultiSelectMustBe0NoneWhenStyleIs1CheckboxS"}

You tried to set the **MultiSelect** property to something other that 0 (None), when the **Style** property of a **ListBox** control is set to 1 (CheckBox). This is not allowed.

Name 'item' conflicts with existing module, project, or object library

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

This error has the following causes:

- You loaded a Form, User Control , User Document, or Property Page that contains objects with conflicting names.

  This can occur if the file was edited outside of Visual Basic.
- You renamed a form to the same name as the project, another form, or a module.
- You renamed the project to the same name as a form or module.

The OLE client control and OLE embeddings are not allowed on UserControls, UserDocuments, or PropertyPages

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

You tried to paste an OLE client control from the Clipboard onto a UserControl, UserDocument, or PropertyPage. This is not allowed.

Only one property and one event per module can be selected User Interface Default. 'item' already has this attribute. Selecting this attribute will void the previous setting.

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

You tried to set the procedure attribute of a property or event to User Interface Default but it was already set on another property or event. This can occur in the **Procedure Attributes** dialog box.

Only one property or event is allowed to have the User Interface Default attribute set at once. Click **OK** to make the current property or event the User Interface Default or click **Cancel** to void the change.

Only one property per module can bind to DataField. Property 'item' already binds to it. Selecting this attribute will void the previous setting.

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

You tried to set the Data Binding attribute of a property to bind to DataField, but another property has this attribute set. This can occur in the **Procedure Attributes** dialog box.

Only one property in a module is allowed to bind to DataField. Click **OK** to set the current property to bind to DataField or click **Cancel** to void the change.

Project file is read-only

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

A change was attempted that would modify the project file, but the project file is under source code control and is not checked out.

Project 'item' can not be referenced because its type is Standard EXE

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

You tried to reference a project whose Project Type is set to Standard EXE. References to Standard EXE projects are not allowed. This error can occur if the file was modified by an editor other than Visual Basic.

RecordSource property of the associated data control is empty

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

An error has occurred while attempting to retrieve information from the associated data control.

Verify that the data control and any objects that it references are correctly registered. Contact the control's vendor for more information if the problem persists.

Some project properties and/or items have to be changed in this type of project. Check Help for details. Are you sure you want to change the project type?

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

You are trying to change the Project Type in the **Project Properties** dialog to a type that requires different settings for some properties or items. These properties or items need to be changed in order to make the project consistent with its new type. Also, the public property of some items in the project, such as User Controls and Class Modules, may be modified as well, since some Project Types do not allow public items.

Sub Main() doesn't exist

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

Sub Main is designated as the Startup Object in the **Project Properties** dialog, however, the current project doesn't have a Sub procedure named Main in any module.

System Error 'item'

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

Visual Basic encountered an error that was generated by the system or an external component and no other useful information was returned.

The specified error number is returned by the system or external component (usually from an Application Interface call) and is displayed in hexadecimal and decimal format.

The application description can't be more than 2000 characters long

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgTheApplicationDescriptionCantBeMoreThan2000CharactersLongC"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgTheApplicationDescriptionCantBeMoreThan2000CharactersLongS"}

You can't have an application description longer than 2000 characters. Delete some characters from the description. This error may occur if the .VBP file was modified in an editor other than Visual Basic.

The compatible server file 'item' is in use by another project

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

Multiple projects loaded in Visual Basic can't use the same compatible server at the same time. For example, if Project1's compatible server is set to Test.DLL and you try to add Project2, which also has its compatible server set to Test.DLL, you will get this error. This is not allowed.

The copy of this file which might have changes is already opened

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

You are trying to access a file that is already open in another project in the project group. For example, you may have a form that belongs to more than one project in the project group. If that form is displayed for the first project, you will get this error if you try to display it in the second project in the project group.

The file 'item' was not registerable as an ActiveX Component.

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

You tried to register a control or DLL which is not a valid ActiveX component. Make sure that you have the correct control or DLL. Contact the ActiveX component's vendor for more information.

The file 'item' was recognized as a Visual Basic file, but the header was corrupt and the file can't be loaded

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

There was an error in the header of the specified file and it cannot be loaded. This error has the following cause and solution:

- Visual Basic could not load the specified file because the header was corrupt. The header contains lines of code specific to Visual Basic and one or more of the header lines were missing or corrupt.

  Install a backup copy of the specified file or modify the file using a text editor to match the header of an existing file.

Warning Modifying a Visual Basic file in an editor other than Visual Basic may corrupt the file. This is only recommended for advanced users.

The file 'item' is already open

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

This error has the following causes and solution:

- You tried to load two project files with the same name in the same project group.
- You tried to save two projects with the same name in the same project group.

Project files cannot have duplicate names in the same project group. You must change the name of one of the project files.

The file, 'item', is marked as a version not supported by the current version of Visual Basic, and won't be loaded

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

You tried to load a form from Visual Basic version 1.0 or a form from a version greater than this version of Visual Basic. This is not supported.

The Instancing property for Class 'item' cannot be set to Creatable SingleUse for an ActiveX DLL

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

The **Instancing** property for the specified class was set to an invalid setting for an ActiveX DLL. Valid settings for the **Instancing** property for an ActiveX DLL are: 1 - **Private**, 2 - **PublicNotCreatable**, 5 - **MultiUse**, or 6 - **GlobalMultiUse**.

The project file 'item1' contains invalid 'item2' key value

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgTheProjectFileitem1ContainsInvaliditem2KeyValueC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgTheProjectFileitem1ContainsInvaliditem2KeyValueS"}

Visual Basic did not recognize the specified key value in the .VBP file. This error has the following cause and solution:

- The .VBP file contains a value for the key that is invalid. The file may be corrupt.

  Install a backup copy of the specified file or modify the file using a text editor to correct the error.

Warning Modifying a Visual Basic file in an editor other than Visual Basic may corrupt the file. This is only recommended for advanced users.

The project file 'item1' contains invalid 'item2' key value. Valid range is 0 to 'item3'

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgTheProjectFileitem1ContainsInvaliditem2KeyValueValidRangeIs0Toitem3C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgTheProjectFileitem1ContainsInvaliditem2KeyValueValidRangeIs0Toitem3S"}

The key value specified in the .VBP file was out of range. This error has the following cause and solution:

- The .VBP file contains a key value that is not in the specified range. The file may be corrupt. Install a backup copy of the specified file or modify the file using a text editor to correct the error.

Warning Modifying a Visual Basic file in an editor other than Visual Basic may corrupt the file. This is only recommended for advanced users.

The project file 'item1' contains invalid key 'item2'. The project can't be loaded

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgTheProjectFileitem1ContainsInvalidKeyitem2TheProjectCantBeLoadedC"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgTheProjectFileitem1ContainsInvalidKeyitem2TheProjectCantBeLoadedS"}

Visual Basic did not recognize the specified key in the .VBP file. This error has the following cause and solution:

- The .VBP file contains a key that Visual Basic doesn't recognize. The file may be corrupt.
  Install a backup copy of the specified file or modify the file using a text editor to correct the error.

Warning Modifying a Visual Basic file in an editor other than Visual Basic may corrupt the file. This is only recommended for advanced users.

The project file 'item' is corrupt, and can't be loaded

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgTheProjectFileitem1IsCorruptCantBeLoadedC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgTheProjectFileitem1IsCorruptCantBeLoadedS"}

Visual Basic cannot read the project file. This can occur if the file has been modified by an editor outside of Visual Basic. To fix the problem, undo any changes that were made to the file.

The project group file 'item' is read-only

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

You tried to save, add, or remove a project from a project group which was marked as read-only. A project group is read-only if the .VBG to which it belongs has its read-only file attribute set.

The project group file 'item' contains a duplicate value, and can't be loaded

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgTheProjectGroupFileitem1ContainsDuplicateValueCantBeLoadedC"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgTheProjectGroupFileitem1ContainsDuplicateValueCantBeLoadedS"}

The project group file (.VBG file) contains a duplicate entry for the StartupProject attribute. This can occur if the .VBG file was edited outside of Visual Basic. Remove the duplicate entry and reload the project.

The project group file 'item' is corrupt, and can't be loaded

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgTheProjectGroupFileitem1IsCorruptCantBeLoadedC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgTheProjectGroupFileitem1IsCorruptCantBeLoadedS"}

A line in the .VBG file is not in the format: <property>=<value>, the <value> string is too big (>260 chars), or <property> is not equal to **Project** or **StartupProject**. This can occur if the .VBG file was edited outside of Visual Basic.

The project group file 'item' is in an unknown format, and can't be loaded

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgTheProjectGroupFileitem1IsInUnknownFormatCantBeLoadedC"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgTheProjectGroupFileitem1IsInUnknownFormatCantBeLoadedS"}

This can occur if the version number or header in the .VBG file is not correct. For Visual Basic 5.0, the version number should be VBGROUP 5.X, where is X can be 0-9.

The project group file 'item' is missing a required value, and can't be loaded

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgTheProjectGroupFileitem1IsMissingRequiredValueCantBeLoadedC"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgTheProjectGroupFileitem1IsMissingRequiredValueCantBeLoadedS"}

A required value was missing in the .VBG file. This error can occur if the .VBG file was modified outside of Visual Basic.

The project name is too long. Name has been truncated

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

The project name has been truncated because it was too long.

A project with the name 'item' is already loaded

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

You tried to load a project that has the same name as a project that is already loaded. You cannot have two projects with the same name.

The wizard file 'item' is corrupt, and can't be loaded

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

A line in the .VBZ file is not in the format: <property>=<value>, the <value> string is too big (>260 chars), or <property> is not equal to **Wizard** or **Param**. This can occur if the .VBZ file was edited outside of Visual Basic.

The wizard file 'item' is in an unknown format, and can't be loaded

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

This can occur if the version number or header in the .VBZ file is not correct. For Visual Basic 5.0, the version number should be VBWIZARD 5.X, where is X can be 0-9.

The wizard file 'item' is missing a required value, and can't be loaded

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

A required value was missing in the .VBZ file. The Wizard key must be specified. This error can occur if the .VBZ file was modified outside of Visual Basic.

There is a newer version of 'item1' registered. Do you want to upgrade to version 'item2' ?

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgThereIsNewerVersionOfitem1RegisteredDoYouWantToUpgradeToVersionitem2C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgThereIsNewerVersionOfitem1RegisteredDoYouWantToUpgradeToVersionitem2S"}

There is a newer version of the specified ActiveX control registered on your system (in addition to the original one). This message has the following cause:

- You saved a project with an older version of an ActiveX control, then installed a newer version on your system.

  Select **Yes** to load the newer one or **No** to load the original control.

The project 'item1' can not be built because it references project 'item2' that does not have a compatible server set.

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

You cannot build the project because a project did not set a compatible server. For example, if you build Project1, and Project1 references Project2, but Project2 doesn't have a compatible server set, you will get this error. You must set a compatible server on the Components tab of the **Project Properties** dialog box.

The project 'item1' can not be built because it references project 'item2' that has not been built.

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

You tried to build a project that references another project that hasn't been built. For example, if you build Project1, and Project1 references Project2, but Project2 does not have a version compatible component set, you will get this error. This only occurs when the Project2's Version Compatibility setting is set to **Project Compatibility**. You can solve this problem by building Project2, which automatically sets the version compatible component.

This project is referenced from another project. Are you sure you want to remove it?

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

You are trying to remove a project that is referenced by another project in the group. Selecting **Yes** on this dialog will cause Visual Basic to try and reference the disk version instead of the in-memory project. This may cause a missing reference.

This Procedure ID is already assigned to member 'item'. Assigning it to this member will void the previous setting.

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

You tried to set the Procedure ID of an item to the same value as another item. This message occurs in the **Procedure Attributes** dialog box.

Procedure IDs must be unique within a module. Click **OK** to apply the Procedure ID to the current item or **Cancel** to void the change.

Unable to launch Books Online

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

You were not able to launch Books Online. Make sure it is correctly installed and try again.

Unable to run a Control Project

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

You tried to run an ActiveX Control project. This is not allowed. An ActiveX control cannot be run as an independent project.

Unable to write Designer cache file 'item'. Will just use regular files on Load

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

Your ActiveX Designer was not able to write information to a cache file so the designer will load using uncached information.

In order to improve performance the next time you start Visual Basic, an ActiveX Designer writes a cache file. The designer couldn't write to the cache file due to low disk space or invalid permissions on the drive. Make sure you have enough disk space available and that you have write permissions to the drive. If the problem persists, contact the ActiveX Designer's vendor.

Unattended Project Cannot be visible at runtime

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

A project marked for Unattended Execution cannot have any visible user interface elements. This message has the following cause and solution:

- The Unattended flag was manually set in the .VBP file and the project was compiled or executed in the design environment.

  To solve this problem, remove all forms from the project or change the Startup Object in the **Project Properties** dialog box to (none) or Sub Main.

Version 'item3' of 'item1' is not registered. The control will be upgraded to version 'item2'

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgVersionitem3Ofitem1IsNotRegisteredTheControlWillBeUpgradedToVersionitem2C"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgVersionitem3Ofitem1IsNotRegisteredTheControlWillBeUpgradedToVersionitem2S"}

Your project was saved with a version of the specified ActiveX control that is not available on your system. A newer version of the specified ActiveX control is registered on your system, so your project will be loaded using the newer version of this control.

Visual Basic found an instance of 'item' in a binary form file. VBX's cannot be converted from binary form files in the 32-bit version

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgVisualBasicFoundInstanceOfitemInBinaryFormFileVBXsCannotBeConvertedFromBinaryFormFilesIn32bitVersionC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgVisualBasicFoundInstanceOfitemInBinaryFormFileVBXsCannotBeConvertedFromBinaryFormFilesIn32bitVersionS"}

Visual Basic 5.0 cannot convert VBXs in binary form files to the equivalent ActiveX controls.

You may need an older version of Visual Basic to convert the form file from binary to a text format form, then load it into Visual Basic 5.0.

Visual Basic was not able to start up due to an invalid system configuration

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

Your installation of Visual Basic has not been correctly configured. This error has the following causes:

- Visual Basic may have been incorrectly installed or the installation not fully completed.
- System files or files required by Visual Basic may have been moved or deleted.
- The registry may be corrupt.

In most cases, when this error occurs, you will need to reinstall your version of Visual Basic to properly configure it.

You may only load or create projects with two modules (Forms, MDI Forms, Modules, Classes) in the Visual Basic Working Model. Projects may contain an unlimited (system dependant) number of modules in other versions of Visual Basic 5.0.

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

The Visual Basic Working Model allows only two modules (Forms, MDI Forms, Modules, Classes) for each project. In order to increase the capacity of your Visual Basic projects, you must purchase a different version of Visual Basic 5.0 such as the Professional or Enterprise versions.

You specified a command line argument that requires a project, but no project was specified

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

Occurs if you specify a command line argument that requires a project, but don't specify the project name (as in vb5 /make).

There is a compatibility error between the current project and the version-compatible component: 'item'

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

This error occurs if you run into a compatible component error during a command line build.

For more information, search for *version compatible* in the online Help.

An EXE or DLL name can not be specified for /Make if building a project group.

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

You tried to override the default file name of the DLL or EXE being built during a command line build of a project group file. Project groups may have multiple output files so this is not allowed.

Use the /Outdir command line argument to specify the output directory for a project group.

Can't create a UserControl instance on its own designer

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

You can't create an instance of **UserControl** on its own designer. This error has the following cause and solution:

- You tried to place an instance of a UserControl on its own designer. For example, if you created a UserControl and copied it to the Clipboard, and then opened the controls designer and pasted the control onto it, you would get this error.

  You are not allowed to have an instance of a UserControl on its own designer.

Can't have child controls capable of receiving focus on a UserControl whose CanGetFocus property is False

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

A UserControl can't have child controls that are capable of receiving focus when the **CanGetFocus** property is **False**. This error has the following cause and solution:

- You tried to paste a control that can receive focus onto a UserControl that has its **CanGetFocus** property set to **False**. For example, you copied a **TextBox** control to the Clipboard, set the **CanGetFocus** property to **False**, then tried to paste the **TextBox** control on the UserControl.

  You are not allowed to have controls that can receive focus on a UserControl whose **CanGetFocus** property is set to **False**.

Control 'item' does not have the align property, so it cannot be placed directly on the MDI form

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

The specified control doesn't have an **Align** property so it can't be put on an MDI form. This error has the following causes and solution:

- You placed a UserControl on an MDI form, then changed the **Alignable** property of the UserControl to **False**.
- You may have tried to add a control that doesn't have an **Align** property to the form.

Only controls that have an **Align** property can be placed on an MDI form.

Device I/O error: 'item'

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

External devices are sometimes subject to unanticipated errors. This error has the following cause and solution:

- An input or output error occurred while your program was the specified device.

Make sure the device is operating properly, and then retry the operation.

Device unavailable: 'item'

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

This error has the following cause and solution:

- The specified device is not online or doesn't exist.

  Check power to the device and any cables connecting your computer to the device. If you are trying to access a printer over a network, make sure there is a logical connection between your computer and the printer, for example, a connection associating LPT1 with the network printer ID.

Disk full: 'item'

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

This error has the following causes and solution:

- There isn't enough room on the disk to create required files.

Move any unused files to another disk or delete unnecessary files.

File already exists: 'item'

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

This error has the following causes and solutions:

- You used the **Save As** command to save a currently loaded project, but the project name already exists.

Use a different project name if you don't want to replace the other project.

File not found: 'item'

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

The file was not found where specified. This error has the following cause and solution:

- In the development environment, this error occurs if you attempt to open a project or load a text file that doesn't exist.

  Check the spelling of the project name or file name and the path specification.

Help Context ID must be a nonnegative number

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

You can't specify a negative number for the Help Context ID.

MDI Forms are not allowed in an ActiveX DLL or ActiveX Control project

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

You cannot add an MDI Form to an ActiveX DLL or ActiveX Control project.

No creatable public class module detected. Press F1 for more information

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

You must have a public creatable class module when creating a project of this type.

ActiveX EXE, ActiveX DLL, and ActiveX Control project types require that you have a least one public creatable class module.

No public UserControl detected. Press F1 for more information

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

You tried to build an ActiveX Control project but no UserControl was available in the project.

You must have a UserControl in this type of project.

Procedure ID must be a negative number or zero

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

The Procedure ID must be a negative number or zero. This error has the following cause and solution:

- You set the Procedure ID in the **Procedure Attributes** dialog to a number greater than zero. You are only allowed to enter Standard Procedure IDs that are zero or a negative number.

The ActiveX Control can not be used because it defines an event named 'item' which conflicts with Visual Basic

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

An event in the specified ActiveX control conflicts with a Visual Basic Extender event. Visual Basic automatically adds standard events to ActiveX controls that are added to a project.

When creating your own ActiveX control project, events such as GotFocus and SetFocus are automatically created (by the Extender object) so they don't need to be added to your control.

For controls not created in Visual Basic, see the control's vendor for more information

This User Control is private and will not be accessible from other projects. Set public to true to change this

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

A private UserControl is only accessible to the project in which it is in. To access the control in other projects, you must set its **Public** property to **True**.

Unexpected error occurred in code generator or linker

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

The code generator or linker caused an unexpected error. Select **Yes** to display the errors in Notepad.

This error is most commonly caused by low disk space or other disk related issues.

User Documents are not allowed in Standard EXE and User Control project types

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

UserDocuments are not allowed in Standard EXE or ActiveX Control project types.

User Documents are not allowed in Standard EXE and User Control project types. Item(s) not loaded

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

UserDocuments are not allowed in Standard EXE or ActiveX Control project types. The items won't be loaded.

User Documents are not allowed in this type of project. Item(s) has(ve) to be removed. Are you sure you want to change the project type?

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

UserDocuments are not allowed in Standard EXE or ActiveX Control project types. Changing the Project Type will remove the UserDocuments from the project. Choose **Yes** to change the Project Type or **No** to cancel the change.

The declaration in the Compatible ActiveX Component was: 'item'

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

Informational message. The declaration shown is missing from the component you are attempting to compile. It has either been deleted or renamed.

Possible remedies: Click **OK** to return to the **Incompatible Component** dialog. You can then:

- Click **Accept** to accept this version incompatible change.
- Click **Accept All** to accept this and all subsequent version incompatible changes.
- Click **Edit** to change the member name back to what it was, or add the missing member.

Note Accepting even one incompatible change will render this version of your component incompatible with all clients compiled using prior versions. It is strongly recommended that you change the file name and **Project Name** settings, to avoid potentially serious program errors in those clients.

The declaration in the Compatible ActiveX Component was: 'item1' It has been changed to: 'item2'

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

Informational message. Displays the difference between a declaration in the component you are attempting to compile, and the version-compatible component that is selected on the Components Tab of the **Project Properties** dialog box.

Possible remedies: Click **OK** to return to the **Incompatible Component** dialog. You can then:

- Click **Accept** to accept this version incompatible change.
- Click **Accept All** to accept this and all subsequent version incompatible changes.
- Click **Edit** to change the declaration back to what it was.

Note Accepting even one incompatible change will render this version of your component incompatible with all clients compiled using prior versions. It is strongly recommended that you change the file name and **Project Name** settings, to avoid potentially serious program errors in those clients.

Unable to set the version compatible component: 'item'

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

The version-compatible component could not be set. This error has the following cause and solution:

- During the load of your project, the version-compatible component could not be set and the Version Compatibility option in the Components tab of the **Project Properties** dialog has been reset to **No Compatibility**.

  In order for Visual Basic to make sure you have a version-compatible project, you must select a version-compatible component in the Components tab of the **Project Properties** dialog.

The Visual Basic Development Environment can't provide multiple instances of a single use class. Consult the documentation for restrictions on debugging single-use objects

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

You are only allowed one instance of a single-use object when debugging in the development environment.

You are trying to create more than one object marked as single-use from the development environment. Each instance of a single use object creates a new instance of the component. This is not allowed in the development environment.

To allow the creation of more than one instance of the object, set the **Instancing** property of the object to 5 - MultiUse, or compile the component and use the object from a compiled executable.

Multiple copies of the shared file 'item' have been modified. In the following dialog, select the copy that should be saved

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

You have modified a file that is shared between two or more projects. Visual Basic doesn't know which file you want saved. Using the list box in the following dialog, select or deselect the files that you want saved for this project.

AddItem Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthAddItemC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthAddItemX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthAddItemA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthAddItemS"}

Adds an item to a **ListBox** or **ComboBox** control or adds a row to a **MS Flex Grid** control. Doesn't support named arguments.

Syntax

object.**AddItem** *item*, *index*

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
item	Required. string expression specifying the item to add to the object. For the **MS Flex Grid** control only, use the tab character (character code 09) to separate multiple strings you want to insert into each column of a newly added row.
index	Optional. Integer specifying the position within the object where the new item or row is placed. For the first item in a **ListBox** or **ComboBox** control or for the first row in a **MS Flex Grid** control, *index* is 0.

Remarks

If you supply a valid value for *index*, *item* is placed at that position within the *object*. If *index* is omitted, *item* is added at the proper sorted position (if the **Sorted** property is set to **True**) or to the end of the list (if **Sorted** is set to **False**).

A **ListBox** or **ComboBox** control that is bound to a **Data** control doesn't support the **AddItem** method.

Arrange Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthArrangeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthArrangeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthArrangeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthArrangeS"}

Arranges the windows or icons within an **MDIForm** object. Doesn't support named arguments.

Syntax

object.**Arrange** *arrangement*

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
arrangement	Required. A value or constant that specifies how to arrange windows or icons on an **MDIForm** object, as described in Settings.

Settings

The settings for *arrangement* are:

Constant	Value	Description
vbCascade	0	Cascades all nonminimized MDI child forms
vbTileHorizontal	1	Tiles all nonminimized MDI child forms horizontally
vbTileVertical	2	Tiles all nonminimized MDI child forms vertically
vbArrangeIcons	3	Arranges icons for minimized MDI child forms

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

Windows or icons are arranged even if the **MDIForm** object is minimized. Results are visible when the **MDIForm** is maximized.

Clear Method (Clipboard, ComboBox, ListBox)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthClearC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthClearX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthClearA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthClearS"}

Clears the contents of a **ListBox**, **ComboBox**, or the system Clipboard.

Syntax

object.**Clear**

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

Remarks

A **ListBox** or **ComboBox** control bound to a **Data** control doesn't support the **Clear** method.

Cls Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthClsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthClsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthClsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthClsS"}

Clears graphics and text generated at run time from a **Form** or **PictureBox**.

Syntax

object.**Cls**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list. If *object* is omitted, the **Form** with the focus is assumed to be *object*.

Remarks

Cls clears text and graphics generated at run time by graphics and printing statements. Background bitmaps set using the **Picture** property and controls placed on a **Form** at design time aren't affected by **Cls**. Graphics and text placed on a **Form** or **PictureBox** while the **AutoRedraw** property is set to **True** aren't affected if **AutoRedraw** is set to **False** before **Cls** is invoked. That is, you can maintain text and graphics on a **Form** or **PictureBox** by manipulating the **AutoRedraw** property of the object you're working with.

After **Cls** is invoked, the **CurrentX** and **CurrentY** properties of *object* are reset to 0.

Drag Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthDragC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthDragX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthDragA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthDragS"}

Begins, ends, or cancels a drag operation of any control except the **Line**, **Menu**, **Shape**, **Timer**, or **CommonDialog** controls. Doesn't support named arguments.

Syntax

object.**Drag** *action*

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the object whose event procedure contains the **Drag** method is assumed.
action	Optional. A constant or value that specifies the action to perform, as described in Settings. If *action* is omitted, the default is to begin dragging the object.

Settings

The settings for *action* are:

Constant	Value	Description
vbCancel	0	Cancels drag operation
vbBeginDrag	1	Begins dragging *object*
vbEndDrag	2	Ends dragging and drop *object*

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

Using the **Drag** method to control a drag-and-drop operation is required only when the **DragMode** property of the object is set to Manual (0). However, you can use **Drag** on an object whose **DragMode** property is set to Automatic (1 or **vbAutomatic**).

If you want the mouse pointer to change shape while the object is being dragged, use either the **DragIcon** or **MousePointer** property. The **MousePointer** property is only used if no **DragIcon** is specified.

In earlier versions of Visual Basic, **Drag** was an asynchronous method where subsequent statements were invoked even though the Drag action wasn't finished. In Visual Basic version 4.0, **Drag** is a synchronous method in which subsequent statements aren't invoked until the Drag action is finished.

EndDoc Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthEndDocC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthEndDocX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthEndDocA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthEndDocS"}

Terminates a print operation sent to the **Printer** object, releasing the document to the print device or spooler.

Syntax

object.**EndDoc**

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

Remarks

If **EndDoc** is invoked immediately after the **NewPage** method, no additional blank page is printed.

GetData Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthGetDataC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthGetDataX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthGetDataA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthGetDataS"}

Returns a graphic from the **Clipboard** object. Doesn't support named arguments.

Syntax

object.**GetData (***format***)**

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
format	Optional. A constant or value that specifies the **Clipboard** graphics format, as described in Settings. Parentheses must enclose the constant or value. If *format* is 0 or omitted, **GetData** automatically uses the appropriate format.

Settings

The settings for *format* are:

Constant	Value	Description
vbCFBitmap	2	Bitmap (.bmp files)
vbCFMetafile	3	metafile (.wmf files)
vbCFDIB	8	Device-independent bitmap (DIB)
vbCFPalette	9	Color palette

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

If no graphic on the **Clipboard** object matches the expected format, nothing is returned. If only a color palette is present on the **Clipboard** object, a minimum size (1 x 1) DIB is created.

GetFormat Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthGetFormatC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthGetFormatX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthGetFormatA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthGetFormatS"}

Returns an integer indicating whether an item on the **Clipboard** object matches a specified format. Doesn't support named argument.

Syntax

object.**GetFormat (***format***)**

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
format	Required. A value or constant that specifies the **Clipboard** object format, as described in Settings. Parentheses must enclose the constant or value.

Settings

The settings for *format* are:

Constant	Value	Description
vbCFLink	&HBF00	DDE conversation information
vbCFText	1	Text
vbCFBitmap	2	Bitmap (.bmp files)
vbCFMetafile	3	Metafile (.wmf files)
vbCFDIB	8	Device-independent bitmap (DIB)
vbCFPalette	9	Color palette

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

The **GetFormat** method returns **True** if an item on the **Clipboard** object matches the specified format. Otherwise, it returns **False**.

For **vbCFDIB** and **vbCFBitmap** formats, whatever color palette is on the **Clipboard** is used when the graphic is displayed.

GetText Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthGetTextC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthGetTextX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthGetTextA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthGetTextS"}

Returns a text string from the **Clipboard** object. Doesn't support named arguments.

Syntax

object.**GetText (***format***)**

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
format	Optional. A value or constant that specifies the **Clipboard** object format, as described in Settings. Parentheses must enclose the constant or value.

Settings

The settings for *format* are:

Constant	Value	Description
vbCFLink	&HBF00	DDE conversation information
vbCFText	1	(Default) Text
vbCFRTF	&HBF01	Rich Text Format (.rtf file)

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

If no text string on the **Clipboard** object matches the expected format, a zero-length string ("") is returned.

Hide Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthHideC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthHideX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthHideA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthHideS"}

Hides an **MDIForm** or **Form** object but doesn't unload it.

Syntax

object.**Hide**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list. If *object* is omitted, the form with the focus is assumed to be *object*.

Remarks

When a form is hidden, it's removed from the screen and its **Visible** property is set to **False**. A hidden form's controls aren't accessible to the user, but they are available to the running Visual Basic application, to other processes that may be communicating with the application through DDE, and to **Timer** control events.

When a form is hidden, the user can't interact with the application until all code in the event procedure that caused the form to be hidden has finished executing.

If the form isn't loaded when the **Hide** method is invoked, the **Hide** method loads the form but doesn't display it.

KillDoc Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthKillDocC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthKillDocX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthKillDocA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthKillDocS"}

Immediately terminates the current print job.

Syntax

object.**KillDoc**

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

Remarks

If the operating system's Print Manager is handling the print job (the Print Manager is running and has background printing enabled), **KillDoc** deletes the current print job and the printer receives nothing.

If Print Manager isn't handling the print job (background printing isn't enabled), some or all of the data may be sent to the printer before **KillDoc** can take effect. In this case, the printer driver resets the printer when possible and terminates the print job.

LinkExecute Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthLinkExecuteC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthLinkExecuteX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthLinkExecuteA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthLinkExecuteS"}

Sends a command string to the source application in a DDE conversation. Doesn't support named arguments.

Syntax

object.**LinkExecute** *string*

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
string	Required. String expression containing a command recognized by the source application.

Remarks

The actual value of *string* varies depending on the source application. For example, Microsoft Excel and Microsoft Word for Windows accept command strings that consist of their macro commands enclosed in square brackets ([]). To see command strings that a source application accepts, consult documentation for that application.

LinkPoke Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthLinkPokeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthLinkPokeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthLinkPokeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthLinkPokeS"}

Transfers the contents of a **Label**, **PictureBox**, or **TextBox** control to the source application in a DDE conversation.

Syntax

object.**LinkPoke**

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

Remarks

The *object* is the name of a **Label**, **PictureBox**, or **TextBox** involved in a DDE conversation as a destination. If *object* is a **Label**, **LinkPoke** transfers the contents of the **Caption** property to the source. If *object* is a **PictureBox**, **LinkPoke** transfers the contents of the **Picture** property to the source. If *object* is a **TextBox**, **LinkPoke** transfers the contents of the **Text** property to the source.

Typically, information in a DDE conversation flows from source to destination. However, **LinkPoke** allows a destination object to supply data to the source. Not all source applications accept information supplied this way; if the source application doesn't accept the data, an error occurs.

LinkRequest Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthLinkRequestC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthLinkRequestX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthLinkRequestA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthLinkRequestS"}

Asks the source application in a DDE conversation to update the contents of a **Label**, **PictureBox**, or **TextBox** control.

Syntax

object.**LinkRequest**

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

Remarks

The *object* is the name of a **Label**, **PictureBox**, or **TextBox** involved in a DDE conversation as a destination. **LinkRequest** causes the source application to send the most current data to *object*, updating the **Caption** property setting if *object* is a **Label**, the **Picture** property setting if *object* is a **PictureBox**, or the **Text** property setting if *object* is a **TextBox**.

If the **LinkMode** property of *object* is set to Automatic (1 or **vbLinkAutomatic**), the source application automatically updates *object* and **LinkRequest** isn't needed. If the **LinkMode** property of *object* is set to Manual (2 or **vbLinkManual**), the source application updates *object* only when **LinkRequest** is used. If the **LinkMode** property of *object* is set to Notify (3 or **vbLinkNotify**), the source notifies the destination that data has changed by invoking the LinkNotify event. The destination must then use **LinkRequest** to update the data.

LinkSend Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthLinkSendC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthLinkSendX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthLinkSendA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthLinkSendS"}

Transfers the contents of a **PictureBox** control to the destination application in a DDE conversation.

Syntax

object.**LinkSend**

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

Remarks

The *object* must be a **PictureBox** on a **Form** object that is a source in a DDE conversation.

When other applications establish automatic links with a **Form** in your application, Visual Basic notifies them when the contents of a **TextBox** or a **Label** on the **Form** change. However, Visual Basic doesn't automatically notify a DDE destination application when the **Picture** property setting of a **PictureBox** on a source **Form** changes. Because the amount of data in a graphic can be very large and because it seldom makes sense to update a destination application as each pixel in the picture changes, Visual Basic requires that you use the **LinkSend** method to explicitly notify DDE destination applications when the contents of a **PictureBox** changes.

LoadResData Function

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthLoadResDataC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthLoadResDataX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthLoadResDataS"}

Loads data of several possible types from a resource (.res) file and returns a **Byte** array.

Syntax

LoadResData(*index*, *format***)**

The **LoadResData** function syntax has these parts:

Part	Description
index	Required. Integer or string specifying the identifier (ID) of the data in the resource file. The resource whose ID is 1 is reserved for the application icon.
format	Required. Value that specifies the original format of the data being returned, as described in Settings. Value can also be the string name of a user-defined resource.

Settings

The settings for *format* are:

Setting	Description
1	Cursor resource
2	Bitmap resource
3	Icon resource
4	Menu resource
5	Dialog box
6	String resource
7	Font directory resource
8	Font resource
9	Accelerator table
10	User-defined resource
12	Group cursor
14	Group icon

Remarks

The data that **LoadResData** loads from the resource file can be up to 64K.

Using **LoadResData** with a bitmap, icon, or cursor resource type returns a string containing the actual bits in the resource. If you want to use the actual bitmap, icon, or resource, use the **LoadResPicture** function.

Using **LoadResData** is useful for localizing a Visual Basic application because the resources that need to be translated are isolated in one resource file and there is no need to access the source code or recompile the application.

LoadResPicture Function

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthLoadResPictureC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthLoadResPictureX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthLoadResPictureS"}

Loads a bitmap, icon, or cursor from a resource (.res) file.

Syntax

LoadResPicture(*index*, *format***)**

The **LoadResPicture** function syntax has these parts:

Part	Description
index	Required. Integer or string specifying the identifier (ID) of the data in the resource file. The resource whose ID is 1 is reserved for the application icon.
format	Required. Value or constant that specifies the format of the data being returned, as described in Settings.

Settings

The settings for *format* are:

Constant	Value	Description
vbResBitmap	0	Bitmap resource
vbResIcon	1	Icon resource
vbResCursor	2	Cursor resource

Remarks

You can use the **LoadResPicture** function instead of referring to graphics stored in the **Picture** property of a **Form** or controls.

Storing bitmaps, icons, or cursors in and accessing them from resource files improves load time because you can load them individually as needed from the resource file, rather than all at once when a **Form** is loaded.

Using **LoadResPicture** is useful for localizing a Visual Basic application because the resources that need to be translated are isolated in one resource file and there is no need to access the source code or recompile the application.

LoadResString Function

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthLoadResStringC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthLoadResStringX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthLoadResStringS"}

Loads a string from a resource (.res) file.

Syntax

LoadResString(*index***)**

The **LoadResString** function syntax has these parts:

Part	Description
index	Required. Integer specifying the identifier (ID) of the data in the resource file. The resource whose ID is 1 is reserved for the application icon.

Remarks

You can use the **LoadResString** function instead of string literals in your code. Storing long strings of data in and accessing them from resource files improves load time because you can load them individually as needed from the resource file, rather than all at once when a form is loaded.

Using **LoadResString** is useful for localizing a Visual Basic application because the resources that need to be translated are isolated in one resource file and there is no need to access the source code or recompile the application.

Move Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthMoveC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthMoveX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthMoveA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthMoveS"}

Moves an **MDIForm**, **Form**, or control. Doesn't support named arguments.

Syntax

object.**Move** *left*, *top*, *width*, *height*

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

Part	Description
object	Optional. An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the form with the focus is assumed to be *object*.
left	Required. Single-precision value indicating the horizontal coordinate (x-axis) for the left edge of *object*.
top	Optional. Single-precision value indicating the vertical coordinate (y-axis) for the top edge of *object*.
width	Optional. Single-precision value indicating the new width of *object*.
height	Optional. Single-precision value indicating the new height of *object*.

Remarks

Only the *left* argument is required. However, to specify any other arguments, you must specify all arguments that appear in the syntax before the argument you want to specify. For example, you can't specify *width* without specifying *left* and *top*. Any trailing arguments that are unspecified remain unchanged.

For forms and controls in a **Frame** control, the coordinate system is always in twips. Moving a form on the screen or moving a control in a **Frame** is always relative to the origin (0,0), which is the upper-left corner. When moving a control on a **Form** object or in a **PictureBox** (or an MDI child form on an **MDIForm** object), the coordinate system of the container object is used. The coordinate system or unit of measure is set with the **ScaleMode** property at design time. You can change the coordinate system at run time with the **Scale** method.

NewPage Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthNewPageC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthNewPageX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthNewPageA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthNewPageS"}

Ends the current page and advances to the next page on the **Printer** object.

Syntax

object.**NewPage**

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

Remarks

NewPage advances to the next printer page and resets the print position to the upper-left corner of the new page. When invoked, **NewPage** increments the **Printer** object's **Page** property by 1.

PaintPicture Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthPaintPictureMoveC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthPaintPictureX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthPaintPictureA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthPaintPictureS"}

Draws the contents of a graphics file (.bmp, .wmf, .emf, .ico, or .dib) on a **Form**, **PictureBox**, or **Printer**. Doesn't support named arguments.

Syntax

object.**PaintPicture** *picture*, *x1*, *y1*, *width1*, *height1*, *x2*, *y2*, *width2*, *height2*, *opcode*

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

Part	Description
object	Optional. An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the **Form** object with the focus is assumed to be *object*.
Picture	Required. The source of the graphic to be drawn onto *object*. Must be the **Picture** property of a **Form** or **PictureBox**.
x1, *y1*	Required. Single-precision values indicating the destination coordinates (x-axis and y-axis) on *object* for *picture* to be drawn. The **ScaleMode** property of *object* determines the unit of measure used.
Width1	Optional. Single-precision value indicating the destination width of *picture*. The **ScaleMode** property of *object* determines the unit of measure used. If the destination width is larger or smaller than the source width (*width2*), *picture* is stretched or compressed to fit. If omitted, the source width is used.
Height1	Optional. Single-precision value indicating the destination height of *picture*. The **ScaleMode** property of *object* determines the unit of measure used. If the destination height is larger or smaller than the source height (*height2*), *picture* is stretched or compressed to fit. If omitted, the source height is used.
x2, *y2*	Optional. Single-precision values indicating the coordinates (x-axis and y-axis) of a clipping region within *picture*. The **ScaleMode** property of *object* determines the unit of measure used. If omitted, 0 is assumed.
Width2	Optional. Single-precision value indicating the source width of a clipping region within *picture*. The **ScaleMode** property of *object* determines the unit of measure used. If omitted, the entire source width is used.
Height2	Optional. Single-precision value indicating the source height of a clipping region within *picture*. The **ScaleMode** property of *object* determines the unit of measure used. If omitted, the entire source height is used.
Opcode	Optional. Long value or code that is used only with bitmaps. It defines a bit-wise operation (for example, **Not** or **Xor** operator) that is performed on *picture* as it's drawn on *object*. For a complete list of bit-wise operators, see the BitBlt topic in the Windows SDK Help file (Win31wh.hlp).

Remarks

You can flip a bitmap horizontally or vertically by using negative values for the destination height (*height1*) and/or the destination width (*width1*).

You can omit as many optional trailing arguments as you want. If you omit an optional trailing argument or arguments, don't use any commas following the last argument you specify. If you want to specify an optional argument, you must specify all optional arguments that appear in the syntax before it.

Point Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthPointC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthPointX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthPointA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthPointS"}

Returns, as a long integer, the red-green-blue (RGB) color of the specified point on a **Form** or **PictureBox**. Doesn't support named arguments.

Syntax

object.**Point(***x*, *y***)**

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

Part	Description
object	Optional. An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the **Form** object with the focus is assumed to be *object*.
x, *y*	Required. Single-precision values indicating the horizontal (x-axis) and vertical (y-axis) coordinates of the point in the **ScaleMode** property of the **Form** or **PictureBox**. Parentheses must enclose the values.

Remarks

If the point referred to by the *x* and *y* coordinates is outside *object*, the **Point** method returns -1.

PopupMenu Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthPopupMenuC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthPopupMenuX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthPopupMenuA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthPopUpMenuS"}

Displays a pop-up menu on an **MDIForm** or **Form** object at the current mouse location or at specified coordinates. Doesn't support named arguments.

Syntax

object.**PopupMenu** *menuname*, *flags*, *x*, *y, boldcommand*

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

Part	Description
object	Optional. An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the form with the focus is assumed to be *object.*
Menuname	Required. The name of the pop-up menu to be displayed. The specified menu must have at least one submenu.
Flags	Optional. A value or constant that specifies the location and behavior of a pop-up menu, as described in Settings.
X	Optional. Specifies the x-coordinate where the pop-up menu is displayed. If omitted, the mouse coordinate is used.
Y	Optional. Specifies the y-coordinate where the pop-up menu is displayed. If omitted, the mouse coordinate is used.
boldcommand	Optional. Specifies the name of a menu control in the pop-up menu to display its caption in bold text. If omitted, no controls in the pop-up menu appear in bold. This argument works only for applications running under Windows 95. The application will ignore this argument when running under 16-bit versions of Windows or Windows NT 3.51 and earlier.

Settings

The settings for *flags* are:

Constant (location)	Value	Description
vbPopupMenuLeftAlign	0	(Default) The left side of the pop-up menu is located at *x*.
vbPopupMenuCenterAlign	4	The pop-up menu is centered at *x*.
vbPopupMenuRightAlign	8	The right side of the pop-up menu is located at *x*.

Constant (behavior)	Value	Description
vbPopupMenuLeftButton	0	(Default) An item on the pop-up menu reacts to a mouse click only when you use the left mouse button.

vbPopupMenuRightButton	2	An item on the pop-up menu reacts to a mouse click when you use either the right or the left mouse button.

Note The *flags* parameter has no effect on applications running under Microsoft Windows version 3.0 or earlier. To specify two *flags*, combine one constant from each group using the **Or** operator.

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

You specify the unit of measure for the x and y coordinates using the **ScaleMode** property. The x and y coordinates define where the pop-up is displayed relative to the specified form. If the x and y coordinates aren't included, the pop-up menu is displayed at the current location of the mouse pointer.

When you display a pop-up menu, the code following the call to the **PopupMenu** method isn't executed until the user either chooses a command from the menu (in which case the code for that command's Click event is executed before the code following the **PopupMenu** statement) or cancels the menu. In addition, only one pop-up menu can be displayed at a time; therefore, calls to this method are ignored if a pop-up menu is already displayed or if a pull-down menu is open.

PrintForm Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthPrintFormC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthPrintFormx":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthPrintFormA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthPrintFormS"}

Sends a bit-by-bit image of a **Form** object to the printer.

Syntax

object.**PrintForm**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list. If *object* is omitted, the **Form** with the focus is assumed to be *object*.

Remarks

PrintForm prints all visible objects and bitmaps of the **Form** object. **PrintForm** also prints graphics added to a **Form** object or **PictureBox** control at run time if the **AutoRedraw** property is **True** when the graphics are drawn.

The printer used by **PrintForm** is determined by the operating system's Control Panel settings.

RemoveItem Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthRemoveItemC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthRemoveItemX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthRemoveItemA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthRemoveItemS"}

Removes an item from a **ListBox** or **ComboBox** control or a row from a **MS Flex Grid** control. Doesn't support named arguments.

Syntax

object.**RemoveItem** *index*

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
index	Required. Integer representing the position within the object of the item or row to remove. For the first item in a **ListBox** or **ComboBox** or for the first row in a **MS Flex Grid** control, *index* = 0.

Remarks

A **ListBox** or **ComboBox** that is bound to a **Data** control doesn't support the **RemoveItem** method.

Scale Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthScaleC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthScaleEX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthScaleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthScaleS"}

Defines the coordinate system for a **Form**, **PictureBox**, or **Printer**. Doesn't support named arguments.

Syntax

object.**Scale (***x1*, *y1***) - (***x2*, *y2***)**

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

Part	Description
object	Optional. An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the **Form** object with the focus is assumed to be *object.*
x1, *y1*	Optional. Single-precision values indicating the horizontal (x-axis) and vertical (y-axis) coordinates that define the upper-left corner of *object*. Parentheses must enclose the values. If omitted, the second set of coordinates must also be omitted.
x2, *y2*	Optional. Single-precision values indicating the horizontal and vertical coordinates that define the lower-right corner of *object.* Parentheses must enclose the values. If omitted, the first set of coordinates must also be omitted.

Remarks

The **Scale** method enables you to reset the coordinate system to any scale you choose. **Scale** affects the coordinate system for both run-time graphics statements and the placement of controls.

If you use **Scale** with no arguments (both sets of coordinates omitted), it resets the coordinate system to twips.

ScaleX, ScaleY Methods

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthScaleXC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthScaleXX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthScaleXA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthScaleXS"}

Converts the value for the width or height of a **Form**, **PictureBox**, or **Printer** from one of the **ScaleMode** property's unit of measure to another. Doesn't support named arguments.

Syntax

object.**ScaleX (***width*, *fromscale*, *toscale***)**
object.**ScaleY (***height*, *fromscale*, *toscale***)**

The **ScaleX** and **ScaleY** method syntaxes have these parts:

Part	Description
object	Optional. An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the **Form** object with the focus is assumed to be *object*.
width	Required. Specifies, for *object*, the number of units of measure to be converted.
height	Required. Specifies, for *object*, the number of units of measure to be converted.
fromscale	Optional. A constant or value specifying the coordinate system from which *width* or *height* of *object* is to be converted, as described in Settings. The possible values of *fromscale* are the same as for the **ScaleMode** property, plus the new value of HiMetric.
toscale	Optional. A constant or value specifying the coordinate system to which *width* or *height* of *object* is to be converted, as described in Settings. The possible values of *toscale* are the same as for the **ScaleMode** property, plus the new value of HiMetric.

Settings

The settings for *fromscale* and *toscale* are:

Constant	Value	Description
vbUser	0	User-defined: indicates that the width or height of *object* is set to a custom value.
vbTwips	1	Twip (1440 twips per logical inch; 567 twips per logical centimeter).
vbPoints	2	Point (72 points per logical inch).
vbPixels	3	Pixel (smallest unit of monitor or printer resolution).
vbCharacters	4	Character (horizontal = 120 twips per unit; vertical = 240 twips per unit).
vbInches	5	Inch
vbMillimeters	6	Millimeter
vbCentimeters	7	Centimeter
N/A	8	HiMetric. If *fromscale* is omitted, HiMetric is assumed as the default.

Remarks

The **ScaleX** and **ScaleY** methods take a value (*width* or *height*), with its unit of measure specified by *fromscale*, and convert it to the corresponding value for the unit of measure specified by *toscale*.

You can also use **ScaleX** and **ScaleY** with the **PaintPicture** method.

SetData Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthSetDataC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthSetDataX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthSetDataA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthSetDataS"}

Puts a picture on the **Clipboard** object using the specified graphic format. Doesn't support named arguments.

Syntax

object.**SetData** *data*, *format*

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
data	Required. A graphic to be placed on the **Clipboard** object.
format	Optional. A constant or value that specifies one of the **Clipboard** object formats recognized by Visual Basic, as described in Settings. If *format* is omitted, **SetData** automatically determines the graphic format.

Settings

The settings for *format* are:

Constant	Value	Description
vbCFBitmap	2	Bitmap (.bmp files)
vbCFMetafile	3	Metafile (.wmf files)
vbCFDIB	8	Device-independent bitmap (DIB)
vbCFPalette	9	Color palette

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

You set the graphic that is to be placed onto the **Clipboard** object with either the **LoadPicture** function or the **Picture** property of a **Form**, **Image**, or **PictureBox**.

SetText Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthSetTextC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthSetTextX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthSetTextA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthSetTextS"}

Puts a text string on the **Clipboard** object using the specified **Clipboard** object format. Doesn't support named arguments.

Syntax

object.**SetText** *data*, *format*

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
data	Required. String data to be placed onto the Clipboard.
Format	Optional. A constant or value that specifies one of the Clipboard formats recognized by Visual Basic, as described in Settings.

Settings

The settings for *format* are:

Constant	Value	Description
vbCFLink	&HBF00	DDE conversation information
vbCFRTF	&HBF01	RichText Format
vbCFText	1	(Default) Text

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

Show Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthShowC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthShowX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthShowA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthShowS"}

Displays an **MDIForm** or **Form** object. Doesn't support named arguments.

Syntax

object.**Show** *style*

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

Part	Description
object	Optional. An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the form associated with the active form module is assumed to be *object.*
Style	Optional. Integer that determines if the form is modal or modeless. If *style* is 0, the form is modeless; if *style* is 1, the form is modal.

Remarks

If the specified form isn't loaded when the **Show** method is invoked, Visual Basic automatically loads it.

When **Show** displays a modeless form, subsequent code is executed as it's encountered. When **Show** displays a modal form, no subsequent code is executed until the form is hidden or unloaded.

When **Show** displays a modal form, no input (keyboard or mouse click) can occur except to objects on the modal form. The program must hide or unload a modal form (usually in response to some user action) before input to another form can occur. An **MDIForm** can't be modal.

Although other forms in your application are disabled when a modal form is displayed, other applications aren't.

The startup form of an application is automatically shown after its Load event is invoked.

TextHeight Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthTextHeightC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthTextHeightX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthTextHeightA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthTextHeightS"}

Returns the height of a text string as it would be printed in the current font of a **Form**, **PictureBox**, or **Printer**. Doesn't support named arguments.

Syntax

object.**TextHeight(***string***)**

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

Part	Description
object	Optional. An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the **Form** object with the focus is assumed to be *object*.
String	Required. A string expression that evaluates to a string for which the text height is determined. Parentheses must enclose the string expression.

Remarks

The height is expressed in terms of the **ScaleMode** property setting or **Scale** method coordinate system in effect for *object*. Use **TextHeight** to determine the amount of vertical space required to display the text. The height returned includes the normal leading space above and below the text, so you can use the height to calculate and position multiple lines of text within *object*.

If *string* contains embedded carriage returns, **TextHeight** returns the cumulative height of the lines, including the leading space above and below each line.

TextWidth Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthTextWidthC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthTextWidthX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthTextWidthA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthTextWidthS"}

Returns the width of a text string as it would be printed in the current font of a **Form**, **PictureBox**, or **Printer**. Doesn't support named arguments.

Syntax

object.**TextWidth(***string***)**

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

Part	Description
object	Optional. An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the **Form** with the focus is assumed to be *object*.
String	Required. A string expression that evaluates to a string for which the text height is determined. Parentheses must surround the string expression.

Remarks

The width is expressed in terms of the **ScaleMode** property setting or **Scale** method coordinate system in effect for *object*. Use **TextWidth** to determine the amount of horizontal space required to display the text. If *string* contains embedded carriage returns, **TextWidth** returns the width of the longest line.

ZOrder Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthZOrderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthZOrderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthZOrderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthZOrderS"}

Places a specified **MDIForm**, **Form**, or control at the front or back of the z-order within its graphical level. Doesn't support named arguments.

Syntax

object.**ZOrder** *position*

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

Part	Description
object	Optional. An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the form with the focus is assumed to be *object*.
Position	Optional. Integer indicating the position of *object* relative to other instances of the same *object*. If position is 0 or omitted, *object* is positioned at the front of the z-order. If position is 1, *object* is positioned at the back of the z-order.

Remarks

The z-order of objects can be set at design time by choosing the Bring To Front or Send To Back menu command from the Edit menu.

Within an **MDIForm** object, **ZOrder** sends MDI child forms to either the front or the back of the MDI client area, depending on the value of *position*. For an **MDIForm** or **Form** object, **ZOrder** sends the form to either the front or the back of the screen, depending on the value of *position*. As a result, forms can be displayed in front of or behind other running applications.

Three graphical layers are associated with forms and containers. The back layer is the drawing space where the results of the graphics methods are displayed. Next is the middle layer where graphical objects and **Label** controls are displayed. The front layer is where all nongraphical controls like **CommandButton**, **CheckBox**, or **ListBox** are displayed. Anything contained in a layer closer to the front covers anything contained in the layer(s) behind it. **ZOrder** arranges objects only within the layer where the object is displayed.

AddItem Method Example

This example uses the **AddItem** method to add 100 items to a list box. To try this example, paste the code into the Declarations section of a form with a **ListBox** control named List1, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim Entry, I, Msg ' Declare variables.
   Msg = "Choose OK to add 100 items to your list box."
   MsgBox Msg  ' Display message.
   For I = 1 To 100  ' Count from 1 to 100.
      Entry = "Entry " & I ' Create entry.
      List1.AddItem Entry  ' Add the entry.
   Next I
   Msg = "Choose OK to remove every other entry."
   MsgBox Msg  ' Display message.
   For I = 1 To 50   ' Determine how to
      List1.RemoveItem I   ' remove every other
   Next I   ' item.
   Msg = "Choose OK to remove all items from the list box."
   MsgBox Msg  ' Display message.
   List1.Clear ' Clear list box.
End Sub
```

Arrange Method Example

This example uses the **Arrange** method to arrange windows and icons in an MDI form. To try this example, paste the code into the Declarations section of an MDI form named MDIForm1 that has an MDI child form (named Form1, with its **MDIChild** property set to **True**) and a picture box on the MDI Form (named Picture1). Press F5 and click anywhere in the picture box to see the effects of the **Arrange** method.

```
Const FORMCOUNT = 5
Dim F(1 To FORMCOUNT) As New Form1
Private Sub MDIForm_Load ()
   Dim I ' Declare local variable.
   Load Form1   ' Load original Form1.
   For I = 1 To FORMCOUNT
      F(I).Caption = "Form" & I + 1      ' Change caption on copies.
   Next I
End Sub

Private Sub Picture1_Click ()
   Static ClickCount ' Declare variables.
   Dim I, PrevWidth, Start
   ClickCount = ClickCount + 1     ' Increment click counter.
   Select Case ClickCount
      Case 1
         MDIForm1.Arrange 1   ' Tile horizontally.
      Case 2
         MDIForm1.Arrange 2   ' Tile vertically.
      Case 3    ' Minimize each form.
         PrevWidth = MDIForm1.Width      ' Get MDI form width.
         MDIForm1.Width = PrevWidth / 2   ' Divide it in half.
         Form1.WindowState = 1     ' Minimize the original.
         For I = 1 To FORMCOUNT    ' Look at each instance of F.
            F(I).WindowState = 1    ' Minimize each copy of F.
         Next I
         Start = Timer
         Do
         Loop Until Timer = Start + 5
         MDIForm1.Width = PrevWidth      ' Resize to original size.
         MDIForm1.Arrange 3   ' Arrange icons.
   End Select
End Sub
```

Clear Method Example

This example uses the **Clear** method to clear all items from a list box. To try this example, paste the code into the Declarations section of a form with a **ListBox** control named List1, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim Entry, I, Msg ' Declare variables.
   Msg = "Choose OK to add 100 items to your list box."
   MsgBox Msg  ' Display message.
   For I = 1 To 100  ' Count from 1 to 100.
      Entry = "Entry " & I ' Create entry.
      List1.AddItem Entry  ' Add the entry.
   Next I
   Msg = "Choose OK to remove every other entry."
   MsgBox Msg  ' Display message.
   For I = 1 To 50   ' Determine how to
      List1.RemoveItem I   ' remove every other
   Next I   ' item.
   Msg = "Choose OK to remove all items from the list box."
   MsgBox Msg  ' Display message.
   List1.Clear ' Clear list box.
End Sub
```

This example uses the **Clear** method to clear the **Clipboard** object. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Const CF_BITMAP = 2  ' Define bitmap format.
   Dim Msg  ' Declare variable.
   On Error Resume Next ' Set up error handling.
   Msg = "Choose OK to load a bitmap onto the Clipboard."
   MsgBox Msg  ' Display message.
   Clipboard.Clear   ' Clear Clipboard.
   Clipboard.SetData LoadPicture("PAPER.BMP")  ' Get bitmap.
   If Err Then
      Msg = "Can't find the .BMP file."
      MsgBox Msg  ' Display error message.
      Exit Sub
   End If
   Msg = "A bitmap is now on the Clipboard. Choose OK to copy "
   Msg = Msg & "the bitmap from the Clipboard to the form."
   MsgBox Msg  ' Display message.
   Picture = Clipboard.GetData()   ' Copy from Clipboard.
   Msg = "Choose OK to clear the picture."
   MsgBox Msg  ' Display message.
   Picture = LoadPicture() ' Clear picture.
End Sub
```

Cls Method Example

This example uses the **Cls** method to delete printed information from a form. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim Msg  ' Declare variable.
   AutoRedraw = -1   ' Turn on AutoRedraw.
   ForeColor = QBColor(15) ' Set foreground to white.
   BackColor = QBColor(1)  ' Set background to blue.
   FillStyle = 7  ' Set diagonal crosshatch.
   Line (0, 0)-(ScaleWidth, ScaleHeight), , B  ' Put box on form.
   Msg = "This is information printed on the form background."
   CurrentX = ScaleWidth / 2 - TextWidth(Msg) / 2  ' Set X position.
   CurrentY = 2 * TextHeight(Msg)  ' Set Y position.
   Print Msg   ' Print message to form.
   Msg = "Choose OK to clear the information and background "
   Msg = Msg & "pattern just displayed on the form."
   MsgBox Msg  ' Display message.
   Cls   ' Clear form background.
End Sub
```

Drag Method Example

This example uses the **Drag** method to drag the filename of a bitmap (.bmp) file to a picture box where the bitmap is displayed. To try this example, paste all of the code into the Declarations section of a form that contains **DriveListBox**, **DirListBox**, **FileListBox**, **PictureBox**, and **Label** controls. Use the default names for all of the controls. Size and position all controls so they can be easily seen and used. The size and position of the label is unimportant because it's changed at run time. When the program begins, you can browse your file system and load any bitmaps. Once you've located a bitmap that you want to display, click the filename of that bitmap, and drag it to the picture box.

```
Private Sub Form_Load ()
   Picture1.AutoSize = -1  ' Turn on AutoSize.
   Label1.Visible = 0   ' Make the label invisible.
   File1.Pattern = "*.BMP; *.ICO; *.WMF" ' Set file patterns.
End Sub

Private Sub Dir1_Change () ' Any change in Dir1
   File1.Path = Dir1.Path  ' is reflected in File1.
End Sub

Private Sub Drive1_Change () ' Any change in Drive1
   Dir1.Path = Drive1.Drive  ' is reflected in Dir1.
End Sub

Private Sub File1_MouseDown (Button As Integer, Shift As Integer, X As
Single, Y As Single)
   Dim DY   ' Declare variable.
   DY = TextHeight("A") ' Get height of one line.
   Label1.Move File1.Left, File1.Top + Y - DY /2, File1.Width, DY
   Label1.Drag ' Drag label outline.
End Sub

Private Sub Dir1_DragOver (Source As Control, X As Single, Y As Single,
State As Integer)
   ' Change pointer to no drop.
   If State = 0 Then Source.MousePointer = 12
   ' Use default mouse pointer.
   If State = 1 Then Source.MousePointer = 0
End Sub

Private Sub Drive1_DragOver (Source As Control, X As Single, Y As Single,
State As Integer)
   ' Change pointer to no drop.
   If State = 0 Then Source.MousePointer = 12
   ' Use default mouse pointer.
   If State = 1 Then Source.MousePointer = 0
End Sub

Private Sub Form_DragOver (Source As Control, X As Single, Y As Single,
State As Integer)
   ' Change pointer to no drop.
   If State = 0 Then Source.MousePointer = 12
   ' Use default mouse pointer.
   If State = 1 Then Source.MousePointer = 0
End Sub
```

```
Private Sub File1_DragOver (Source As Control, X As Single, Y As Single,
State As Integer)
   On Error Resume Next
   If State = 0 And Right$(File1.Filename,4) = ".ICO" Then
      Label1.DragIcon = LoadPicture(File1.Path + "\" + File1.Filename)
   If Err Then MsgBox "The icon file can't be loaded."
   ElseIf State = 1 Then
      Label1.DragIcon = LoadPicture ()   ' Use no drag icon.
   End If
End Sub

Private Sub Picture1_DragDrop (Source As Control, X As Single, Y As Single)
   On Error Resume Next
   Picture1.Picture = LoadPicture(File1.Path + "\" + File1.Filename)
   If Err Then MsgBox "The picture file can't be loaded."
End Sub
```

EndDoc Method Example

This example uses the **EndDoc** method to end a document after printing two pages, each with a centered line of text indicating the page number. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim HWidth, HHeight, I, Msg     ' Declare variables.
   On Error GoTo ErrorHandler      ' Set up error handler.
   Msg = "This is printed on page"
   For I = 1 To 2 ' Set up two iterations.
      HWidth = Printer.TextWidth(Msg) / 2      ' Get half width.
      HHeight = Printer.TextHeight(Msg) /2     ' Get half height.
      Printer.CurrentX = Printer.ScaleWidth / 2 - HWidth
      Printer.CurrentY = Printer.ScaleHeight / 2 - HHeight
      Printer.Print Msg & Printer.Page & "."   ' Print.
      Printer.NewPage   ' Send new page.
   Next I
   Printer.EndDoc ' Printing is finished.
   Msg = "Two pages, each with a single, centered line of text, "
   Msg = Msg & "have been sent to your printer."
   MsgBox Msg  ' Display message.
   Exit Sub
ErrorHandler:
   MsgBox "There was a problem printing to your printer."
   Exit Sub
End Sub
```

GetData Method Example

This example uses the **GetData** method to copy a bitmap from the **Clipboard** object to a form. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Const CF_BITMAP = 2  ' Define bitmap format.
   Dim Msg  ' Declare variable.
   On Error Resume Next ' Set up error handling.
   Msg = "Choose OK to load a bitmap onto the Clipboard."
   MsgBox Msg  ' Display message.
   Clipboard.Clear   ' Clear Clipboard.
   Clipboard.SetData LoadPicture("PAPER.BMP")  ' Get bitmap.
   If Err Then
      Msg = "Can't find the .bmp file."
      MsgBox Msg  ' Display error message.
      Exit Sub
   End If
   Msg = "A bitmap is now on the Clipboard. Choose OK to copy "
   Msg = Msg & "the bitmap from the Clipboard to the form "
   MsgBox Msg  ' Display message.
   Picture = Clipboard.GetData()   ' Copy from Clipboard.
   Msg = "Choose OK to clear the form."
   MsgBox Msg  ' Display message.
   Picture = LoadPicture() ' Clear form.
End Sub
```

GetFormat Method Example

This example uses the **GetFormat** method to determine the format of the data on the **Clipboard** object. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   ' Define bitmap formats.
   Dim ClpFmt, Msg   ' Declare variables.
   On Error Resume Next ' Set up error handling.
   If Clipboard.GetFormat(vbCFText) Then ClpFmt = ClpFmt + 1
   If Clipboard.GetFormat(vbCFBitmap) Then ClpFmt = ClpFmt + 2
   If Clipboard.GetFormat(vbCFDIB) Then ClpFmt = ClpFmt + 4
   If Clipboard.GetFormat(vbCFRTF) Then ClpFmt = ClpFmt + 8
   Select Case ClpFmt
      Case 1
         Msg = "The Clipboard contains only text."
      Case 2, 4, 6
         Msg = "The Clipboard contains only a bitmap."
      Case 3, 5, 7
         Msg = "The Clipboard contains text and a bitmap."
      Case 8, 9
         Msg = "The Clipboard contains only rich text."
      Case Else
         Msg = "There is nothing on the Clipboard."
   End Select
   MsgBox Msg  ' Display message.
End Sub
```

GetText Method Example

This example uses the **GetText** method to copy a text string from the **Clipboard** object to a string variable. To try this example, paste the code into the Declarations section of a form with a **TextBox** control named Text1, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim I, Msg, Temp  ' Declare variables.
   On Error Resume Next ' Set up error handling.
   Msg = "Type anything you like into the text box below."
   Text1.Text = InputBox(Msg)  ' Get text from user.
   Msg = "Choose OK to copy the contents of the text box "
   Msg = Msg & "to the Clipboard."
   MsgBox Msg  ' Display message.
   Clipboard.Clear   ' Clear Clipboard.
   Clipboard.SetText Text1.Text  ' Put text on Clipboard.
   If Clipboard.GetFormat(vbCFText) Then
      Text1.Text = ""   ' Clear the text box.
      Msg = "The text is now on the Clipboard. Choose OK "
      Msg = Msg & "to copy the text from the Clipboard back "
      Msg = Msg & "to the text box."
      MsgBox Msg  ' Display message.
      Temp = Clipboard.GetText(vbCFText) ' Get Clipboard text.
      For I = Len(Temp) To 1 Step -1     ' Reverse the text.
         Text1.Text = Text1.Text & Mid(Temp, I, 1)
      Next I
   Else
      Msg = "There is no text on the Clipboard."
      MsgBox Msg  ' Display error message.
   End If
End Sub
```

Hide Method Example

This example uses the **Hide** method to hide a form. To try this example, paste the code into the Declarations section of a non-MDI form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim Msg  ' Declare variable.
   Hide  ' Hide form.
   Msg = "Choose OK to make the form reappear."
   MsgBox Msg  ' Display message.
   Show  ' Show form again.
End Sub
```

KillDoc Method Example

This example uses the **KillDoc** method to terminate the current print job. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click()
   For i = 1 To 40
      Printer.CurrentX = 1440      ' Set left margin.
      Printer.CurrentY = (i * 300) ' Advance page to next line.
      Printer.Print "This is line" & Str$(i) & " of text."
      On Error Resume Next ' Catch any printer error.
      If i = 26 Then
         Printer.KillDoc   ' Terminate print job abruptly.
         Printer.EndDoc
         End
      End If
   Next i
End Sub
```

LinkExecute Method Example

This example establishes a DDE link with Microsoft Excel, places some values into cells in the first row of a new worksheet, and charts the values. **LinkExecute** sends Microsoft Excel the command to activate a worksheet, select some values, and chart them. To try this example, Microsoft Excel must be installed on your computer and in the path statement of your Autoexec.bat file. Paste the code into the Declarations section of a form that has a **TextBox** control with the default name Text1, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim Cmd, I, Q, Row, Z   ' Declare variables.
   Q = Chr(34) ' Define quotation marks.
   ' Create a string containing Microsoft Excel macro commands.
   Cmd = "[ACTIVATE(" & Q &"SHEET1" & Q & ")]"
   Cmd = Cmd & "[SELECT(" & Q & "R1C1:R5C2" & Q & ")]"
   Cmd = Cmd & "[NEW(2,1)][ARRANGE.ALL()]"
   If Text1.LinkMode = vbNone Then
      Z = Shell("Excel", 4)  ' Start Microsoft Excel.
      Text1.LinkTopic = "Excel|Sheet1"   ' Set link topic.
      Text1.LinkItem = "R1C1"      ' Set link item.
      Text1.LinkMode = vbLinkManual      ' Set link mode.
   End If
   For I = 1 To 5
      Row = I  ' Define row number.
      Text1.LinkItem = "R" & Row & "C1"  ' Set link item.
      Text1.Text = Chr(64 + I)     ' Put value in Text.
      Text1.LinkPoke ' Poke value to cell.
      Text1.LinkItem = "R" & Row & "C2"  ' Set link item.
      Text1.Text = Row  ' Put value in Text.
      Text1.LinkPoke ' Poke value to cell.
   Next I
   On Error Resume Next
   Text1.LinkExecute Cmd   ' Carry out Microsoft Excel commands.
   MsgBox "LinkExecute DDE demo with Microsoft Excel finished.", 64
   End
End Sub
```

LinkPoke Method Example

This example establishes a DDE link with Microsoft Excel, places some values into cells in the first row of a new worksheet, and charts the values. **LinkPoke** sends the values to be charted to the Microsoft Excel worksheet. To try this example, Microsoft Excel must be installed and in the path statement of your Autoexec.bat file. Paste the code into the Declarations section of a form that has a **TextBox** control with the default name Text1, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim Cmd, I, Q, Row, Z   ' Declare variables.
   Q = Chr(34) ' Define quotation marks.
   ' Create a string containing Microsoft Excel macro commands.
   Cmd = "[ACTIVATE(" & Q &"SHEET1" & Q & ")]"
   Cmd = Cmd & "[SELECT(" & Q & "R1C1:R5C2" & Q & ")]"
   Cmd = Cmd & "[NEW(2,1)][ARRANGE.ALL()]"
   If Text1.LinkMode = vbNone Then
      Z = Shell("Excel", 4)  ' Start Microsoft Excel.
      Text1.LinkTopic = "Excel|Sheet1"   ' Set link topic.
      Text1.LinkItem = "R1C1"      ' Set link item.
      Text1.LinkMode = vbLinkManual      ' Set link mode.
   End If
   For I = 1 To 5
      Row = I  ' Define row number.
      Text1.LinkItem = "R" & Row & "C1"  ' Set link item.
      Text1.Text = Chr(64 + I)     ' Put value in Text.
      Text1.LinkPoke ' Poke value to cell.
      Text1.LinkItem = "R" & Row & "C2"  ' Set link item.
      Text1.Text = Row  ' Put value in Text.
      Text1.LinkPoke ' Poke value to cell.
   Next I
   Text1.LinkExecute Cmd   ' Carry out Microsoft Excel commands.
   On Error Resume Next
   MsgBox "LinkPoke DDE demo with Microsoft Excel finished.", 64
   End
End Sub
```

LinkRequest Method Example

This example uses **LinkRequest** to update the contents of a text box with the values in a Microsoft Excel worksheet. To try this example, you must have Microsoft Excel running on your computer. Place some data in the first cells in the first column in the default worksheet (Sheet1.xls). Paste the code into the Declarations section of a form that has a **TextBox** control called Text1, and then press F5 and click the form.

```
Private Sub Form_Click ()
   If Text1.LinkMode = vbNone Then ' Test link mode.
      Text1.LinkTopic = "Excel|Sheet1"   ' Set link topic.
      Text1.LinkItem = "R1C1"      ' Set link item.
      Text1.LinkMode = vbLinkManual      ' Set link mode.
      Text1.LinkRequest ' Update text box.
   Else
      If Text1.LinkItem  = "R1C1" Then
         Text1.LinkItem = "R2C1"
         Text1.LinkRequest ' Update text box.
      Else
         Text1.LinkItem = "R1C1"
         Text1.LinkRequest ' Update text box.
      End If
   End If
End Sub
```

Move Method Example

This example uses the **Move** method to move a form around on the screen. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim Inch, Msg  ' Declare variables.
   Msg = "Choose OK to resize and move this form by "
   Msg = Msg & "changing the value of properties."
   MsgBox Msg  ' Display message.
   Inch = 1440 ' Set inch in twips.
   Width = 4 * Inch  ' Set width.
   Height = 2 * Inch ' Set height.
   Left = 0 ' Set left to origin.
   Top = 0  ' Set top to origin.
   Msg = "Now choose OK to resize and move this form "
   Msg = Msg & "using the Move method."
   MsgBox Msg  ' Display message.
   Move Screen.Width - 2 * Inch, Screen.Height - Inch, 2 * Inch, Inch
End Sub
```

NewPage Method Example

This example uses the **NewPage** method to begin a new printer page after printing a single, centered line of text on a page. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim HWidth, HHeight, I, Msg     ' Declare variables.
   On Error GoTo ErrorHandler      ' Set up error handler.
   Msg = "This is printed on page"
   For I = 1 To 2 ' Set up two iterations.
      HWidth = Printer.TextWidth(Msg) / 2      ' Get one-half width.
      HHeight = Printer.TextHeight(Msg) /2     ' Get one-half height.
      Printer.CurrentX = Printer.ScaleWidth / 2 - HWidth
      Printer.CurrentY = Printer.ScaleHeight / 2 - HHeight
      Printer.Print Msg & Printer.Page & "."   ' Print.
      Printer.NewPage   ' Send new page.
   Next I
   Printer.EndDoc ' Print done.
   Msg = "Two pages, each with a single, centered line of text, "
   Msg = Msg & "have been sent to your printer."
   MsgBox Msg  ' Display message.
   Exit Sub
ErrorHandler:
   MsgBox "There was a problem printing to your printer."
   Exit Sub
End Sub
```

Point Method Example

This example uses the **Point** method to determine the color of a specific point on a form. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim LeftColor, MidColor, Msg, RightColor    ' Declare variables.
   AutoRedraw = -1   ' Turn on AutoRedraw.
   Height = 3 * 1440 ' Set height to 3 inches.
   Width = 5 * 1440  ' Set width to 5 inches.
   BackColor = QBColor(1)  ' Set background to blue.
   ForeColor = QBColor(4)  ' Set foreground to red.
   Line (0, 0)-(Width / 3, Height), , BF ' Red box.
   ForeColor = QBColor(15) ' Set foreground to white.
   Line (Width / 3, 0)-((Width / 3) * 2, Height), , BF
   LeftColor = Point(0, 0) ' Find color of left box,
   MidColor = Point(Width / 2, Height / 2)     ' middle box, and
   RightColor = Point(Width, Height)     ' right box.
   Msg = "The color number for the red box on the left side of "
   Msg = Msg & "the form is " & LeftColor & ". The "
   Msg = Msg & "color of the white box in the center is "
   Msg = Msg & MidColor & ". The color of the blue "
   Msg = Msg & "box on the right is " & RightColor & "."
   MsgBox Msg  ' Display message.
End Sub
```

PopupMenu Method Example

This example displays a pop-up menu at the cursor location when the user clicks the right mouse button over a form. To try this example, create a form that includes a **Menu** control named mnuFile (mnuFile must have at least one submenu). Copy the code into the Declarations section of the form, and press F5.

```
Private Sub Form_MouseDown (Button As Integer, Shift As Integer, X As
Single, Y As Single)
   If Button = 2 Then
      PopupMenu mnuFile
   End If
End Sub
```

PrintForm Method Example

This example uses the **PrintForm** method to print the current form. To try this example, paste the code into the Declarations section of a form. Place on the form any controls you want to see on the printed form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim Msg  ' Declare variable.
   On Error GoTo ErrorHandler      ' Set up error handler.
   PrintForm   ' Print form.
   Exit Sub
ErrorHandler:
   Msg = "The form can't be printed."
   MsgBox Msg  ' Display message.
   Resume Next
End Sub
```

RemoveItem Method Example

This example uses the **RemoveItem** method to remove entries from a list box. To try this example, paste the code into the Declarations section of a form with a **ListBox** control named List1, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim Entry, I, Msg ' Declare variables.
   Msg = "Choose OK to add 100 items to your list box."
   MsgBox Msg  ' Display message.
   For I = 1 To 100  ' Count from 1 to 100.
      Entry = "Entry " & I ' Create entry.
      List1.AddItem Entry  ' Add the entry.
   Next I
   Msg = "Choose OK to remove every other entry."
   MsgBox Msg  ' Display message.
   For I = 1 To 50   ' Determine how to
      List1.RemoveItem I   ' remove every other
   Next I   ' item.
   Msg = "Choose OK to remove all items from the list box."
   MsgBox Msg  ' Display message.
   List1.Clear ' Clear list box.
End Sub
```

Scale Method Example

This example uses the **Scale** method to set up a custom coordinate system so that a bar chart can be drawn on a form. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim I, OldFontSize   ' Declare variables.
   Width = 8640: Height = 5760     ' Set form size in twips.
   Move 100,100   ' Move form origin.
   AutoRedraw = -1   ' Turn on AutoRedraw.
   OldFontSize = FontSize  ' Save old font size.
   BackColor = QBColor(7)  ' Set background to gray.
   Scale (0, 110)-(130, 0) ' Set custom coordinate system.
   For I = 100 To 10 Step -10
      Line (0, I)-(2, I)   ' Draw scale marks every 10 units.
      CurrentY = CurrentY + 1.5    ' Move cursor position.
      Print I  ' Print scale mark value on left.
      Line (ScaleWidth - 2, I)-(ScaleWidth, I)
      CurrentY = CurrentY + 1.5    ' Move cursor position.
      CurrentX = ScaleWidth - 9
      Print I  ' Print scale mark value on right.
   Next I
   ' Draw bar chart.
   Line (10, 0)-(20, 45), RGB(0, 0, 255), BF   ' First blue bar.
   Line (20, 0)-(30, 55), RGB(255, 0, 0), BF   ' First red bar.
   Line (40, 0)-(50, 40), RGB(0, 0, 255), BF
   Line (50, 0)-(60, 25), RGB(255, 0, 0), BF
   Line (70, 0)-(80, 35), RGB(0, 0, 255), BF
   Line (80, 0)-(90, 60), RGB(255, 0, 0), BF
   Line (100, 0)-(110, 75), RGB(0, 0, 255), BF
   Line (110, 0)-(120, 90), RGB(255, 0, 0), BF
   CurrentX = 18: CurrentY = 100   ' Move cursor position.
   FontSize = 14  ' Enlarge font for title.
   Print "Widget Quarterly Sales"  ' Print title.
   FontSize = OldFontSize  ' Restore font size.
   CurrentX = 27: CurrentY = 93    ' Move cursor position.
   Print "Planned Vs. Actual"      ' Print subtitle.
   Line (29, 86)-(34, 88), RGB(0, 0, 255), BF  ' Print legend.
   Line (43, 86)-(49, 88), RGB(255, 0, 0), BF
End Sub
```

SetText Method Example

This example uses the **SetText** method to copy text from a text box to the Clipboard. To try this example, paste the code into the Declarations section of a form with a text box named Text1, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Const CF_TEXT = 1 ' Define bitmap format.
   Dim I, Msg, Temp  ' Declare variables.
   On Error Resume Next ' Set up error handling.
   Msg = "Type anything you like into the text box below."
   Text1.Text = InputBox(Msg)      ' Get text from user.
   Msg = "Choose OK to copy the contents of the text box "
   Msg = Msg & "to the Clipboard."
   MsgBox Msg  ' Display message.
   ClipBoard.Clear   ' Clear Clipboard.
   Clipboard.SetText Text1.Text    ' Put text on Clipboard.
   If Clipboard.GetFormat(CF_TEXT) Then
      Text1.Text = ""   ' Clear the text box.
      Msg = "The text is now on the Clipboard. Choose OK "
      Msg = Msg & "to copy the text from the Clipboard back "
      Msg = Msg & "to the text box."
      MsgBox Msg  ' Display message.
      Temp = Clipboard.GetText(CF_TEXT)  ' Get Clipboard text.
      For I = Len(Temp) To 1 Step -1     ' Reverse the text.
         Text1.Text = Text1.Text & Mid(Temp, I, 1)
      Next I
   Else
      Msg = "There is no text on the Clipboard."
      MsgBox Msg  ' Display error message.
   End If
End Sub
```

Show Method Example

This example uses the **Show** method to show a hidden form. To try this example, paste the code into the Declarations section of a non-MDI form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim Msg  ' Declare variable.
   Hide  ' Hide form.
   Msg = "Choose OK to make the form reappear."
   MsgBox Msg  ' Display message.
   Show  ' Show form again.
End Sub
```

TextHeight Method Example

The **TextHeight** method is used to center a line of text vertically on a form. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim HalfWidth, HalfHeight, Msg  ' Declare variable.
   AutoRedraw = -1   ' Turn on AutoRedraw.
   BackColor = QBColor(4)  ' Set background color.
   ForeColor = QBColor(15) ' Set foreground color.
   Msg = "Visual Basic" ' Create message.
   FontSize = 48  ' Set font size.
   HalfWidth =  TextWidth(Msg) / 2       ' Calculate one-half width.
   HalfHeight = TextHeight(Msg) / 2      ' Calculate one-half height.
   CurrentX = ScaleWidth / 2 - HalfWidth ' Set X.
   CurrentY = ScaleHeight / 2 - HalfHeight     ' Set Y.
   Print Msg   ' Print message.
End Sub
```

TextWidth Method Example

The **TextWidth** method is used to center a line of text horizontally on a form. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim HalfHeight, HalfWidth, Msg  ' Declare variables.
   AutoRedraw = -1   ' Turn on AutoRedraw.
   BackColor = QBColor(4)  ' Set background color.
   ForeColor = QBColor(15) ' Set foreground color.
   Msg = "Visual Basic" ' Create message.
   FontSize = 48  ' Set font size.
   HalfWidth = TextWidth(Msg) / 2  ' Calculate one-half width.
   HalfHeight = TextHeight(Msg) / 2      ' Calculate one-half height.
   CurrentX = ScaleWidth / 2 - HalfWidth ' Set X.
   CurrentY = ScaleHeight / 2 - HalfHeight     ' Set Y.
   Print Msg   ' Print message.
End Sub
```

App Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjAppC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjAppX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjAppP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjAppM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjAppE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjAppS"}

The **App** object is a global object accessed with the **App** keyword. It determines or specifies information about the application's title, version information, the path and name of its executable file and Help files, and whether or not a previous instance of the application is running.

Syntax

App

Remarks

The **App** object has no events or methods.

CheckBox Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjCheckBoxC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjCheckBoxX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjCheckBoxP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjCheckBoxM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjCheckBoxE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjCheckBoxS"}

A **CheckBox** control displays an X when selected; the X disappears when the **CheckBox** is cleared. Use this control to give the user a True/False or Yes/No option. You can use **CheckBox** controls in groups to display multiple choices from which the user can select one or more. You can also set the value of a **CheckBox** programmatically with the Value property.

Syntax

CheckBox

Remarks

CheckBox and **OptionButton** controls function similarly but with an important difference: Any number of **CheckBox** controls on a form can be selected at the same time. In contrast, only one **OptionButton** in a group can be selected at any given time.

To display text next to the **CheckBox**, set the **Caption** property. Use the **Value** property to determine the state of the control—selected, cleared, or unavailable.

ComboBox Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjComboBoxC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjComboBoxX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjComboBoxP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjComboBoxM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjComboBoxE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjComboBoxS"}

A **ComboBox** control combines the features of a **TextBox** control and a **ListBox** control—users can enter information in the text box portion or select an item from the list box portion of the control.

Syntax

ComboBox

Remarks

To add or delete items in a **ComboBox** control, use the **AddItem** or **RemoveItem** method. Set the **List**, **ListCount**, and **ListIndex** properties to enable a user to access items in the **ComboBox**. Alternatively, you can add items to the list by using the **List** property at design time.

CommandButton Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjCommandButtonC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjCommandButtonX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjCommandButtonP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjCommandButtonM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjCommandButtonE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjCommandButtonS"}

Use a **CommandButton** control to begin, interrupt, or end a process. When chosen, a **CommandButton** appears pushed in and so is sometimes called a push button.

Syntax

CommandButton

Remarks

To display text on a **CommandButton** control, set its **Caption** property. A user can always choose a **CommandButton** by clicking it. To allow the user to choose it by pressing ENTER, set the **Default** property to **True**. To allow the user to choose the button by pressing ESC, set the **Cancel** property of the **CommandButton** to **True**.

Controls Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjControlsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vamthItem;vbobjControlsP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vamthClear;vamthRemove;vbobjControlsM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjControlsE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjControlsS"}

A collection whose elements represent each control on a form, including elements of control array. The **Controls** collection has a single property, **Count**, that specifies the number of elements in an array.

Syntax

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

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

Part	Description
object	An object expression that evaluates to a **Form** object.
Index	An integer with a range from 0 to `Controls.Count - 1`.

Remarks

The **Controls** collection enumerates loaded controls on a form and is useful for iterating through them. The **Controls** collection identifies an intrinsic form-level variable named **Controls**. If you omit the optional *object* placeholder, you must include the **Controls** keyword. However, if you include *object*, you can omit the **Controls** keyword. For example, the following two lines of code have the same effect:

```
MyForm.Controls(6).Top = MyForm.Controls(5).Top + increment
MyForm(6).Top = MyForm(5).Top + increment
```

You can pass **Controls(***index***)** to a function whose argument is specified as a **Controls** class. You can also access members using their name. For example:

```
Controls("Command1").Top
```

DirListBox Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjDirListBoxC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjDirListBoxX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjDirListBoxP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjDirListBoxM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjDirListBoxE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjDirListBoxS"}

A **DirListBox** control displays directories and paths at run time. Use this control to display a hierarchical list of directories. You can create dialog boxes that, for example, enable a user to open a file from a list of files in all available directories.

Syntax

DirListBox

Remarks

Set the **List**, **ListCount**, and **ListIndex** properties to enable a user to access items in a list. If you also display the **DriveListBox** and **FileListBox** controls, you can write code to synchronize them with the **DirListBox** control and with each other.

DriveListBox Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjDriveListBoxC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjDriveListBoxX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjDriveListBoxP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjDriveListBoxM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjDriveListBoxE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjDriveListBoxS"}

A **DriveListBox** control enables a user to select a valid disk drive at run time. Use this control to display a list of all the valid drives in a user's system. You can create dialog boxes that enable the user to open a file from a list of files on a disk in any available drive.

Syntax

DriveListBox

Remarks

Set the **List**, **ListCount**, and **ListIndex** properties to enable a user to access items in the list. If you also display the **DirListBox** and **FileListBox** controls, you can write code to synchronize them with the **DriveListBox** control and with each other.

FileListBox Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjFileListBoxC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjFileListBoxX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjFileListBoxP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjFileListBoxM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjFileListBoxE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjFileListBoxS"}

A **FileListBox** control locates and lists files in the directory specified by the **Path** property at run time. Use this control to display a list of files selected by file type. You can create dialog boxes in your application that, for example, enable the user to select a file or group of files.

Syntax

FileListBox

Remarks

Set the **List**, **ListCount**, and **ListIndex** properties to enable a user to access items in the list. If you also display the **DirListBox** and **DriveListBox** controls, you can write code to synchronize them with the **FileListBox** control and with each other.

Form Object, Forms Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjFormC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjFormsX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaproStartUpPosition;vbobjFormP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjFormM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjFormE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjFormS"}

A **Form** object is a window or dialog box that makes up part of an application's user interface.

A **Forms** collection is a collection whose elements represent each loaded form in an application. The collection includes the application's MDI form, MDI child forms, and non-MDI forms. The **Forms** collection has a single property, **Count**, that specifies the number of elements in the collection.

Syntax

Form
Forms(*index***)**

The placeholder *index* represents an integer with a range from 0 to `Forms.Count - 1`.

Remarks

You can use the **Forms** collection to iterate through all loaded forms in an application. It identifies an intrinsic global variable named **Forms**. You can pass **Forms**(*index*) to a function. whose argument is specified as a **Forms** class.

Forms have properties that determine aspects of their appearance, such as position, size, and color; and aspects of their behavior, such as whether or not they are resizable.

Forms can also respond to events initiated by a user or triggered by the system. For example, you could write code in a form's Click event procedure that would enable the user to change the color of a form by clicking it.

In addition to properties and events, you can use methods to manipulate forms using code. For example, you can use the **Move** method to change a form's location and size.

A special kind of form, the MDI form, can contain other forms called MDI child forms. An MDI form is created with the MDI Form command on the Insert menu; an MDI child form is created by choosing New Form from the File menu and then setting the **MDIChild** property to **True**.

You can create multiple instances of forms in code by using the **New** keyword in **Dim**, **Set**, and **Static** statements.

When designing forms, set the **BorderStyle** property to define a form's border, and set the **Caption** property to put text in the title bar. In code, you can use the **Hide** and **Show** methods to make forms invisible or visible at run time.

Note Setting **BorderStyle** to 0 removes the border. If you want your form to have a border without the title bar, Control-menu box, Maximize button, and Minimize button, delete any text from the form's **Caption** property, and set the form's **ControlBox**, **MaxButton**, and **MinButton** properties to **False**.

Form is an Object data type. You can declare variables as type **Form** before setting them to an instance of a type of form that was declared at design time. Similarly, you can pass an argument to a procedure as type **Form**.

Forms also can act as sources in a DDEconversation, with a **Label**, **PictureBox**, or **TextBox** control furnishing the data.

You can access the collection of controls on a **Form** using the **Controls** collection. For example, to hide all the controls on an **Form** you can use code similar to the following:

```
For Each Control in Form1.Controls
   Control.Visible = False
```

Next Control

Frame Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjFrameC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjFrameX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjFrameP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjFrameM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjFrameE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjFrameS"}

A **Frame** control provides an identifiable grouping for controls. You can also use a **Frame** to subdivide a form functionally—for example, to separate groups of **OptionButton** controls.

Syntax

Frame

Remarks

To group controls, first draw the **Frame** control, and then draw the controls inside the **Frame**. This enables you to move the **Frame** and the controls it contains together. If you draw a control outside the **Frame** and then try to move it inside, the control will be on top of the **Frame** and you'll have to move the **Frame** and controls separately.

To select multiple controls in a **Frame**, hold down the CTRL key while using the mouse to draw a box around the controls.

HScrollBar, VScrollBar Controls

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjHScrollBarC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjHScrollBarX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjHScrollBarP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjHScrollBarM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjHScrollBarE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjHScrollBarS"}

Scroll bars provide easy navigation through a long list of items or a large amount of information. They can also provide an analog representation of current position. You can use a scroll bar as an input device or indicator of speed or quantity—for example, to control the volume of a computer game or to view the time elapsed in a timed process.

Syntax

HScrollBar
VScrollBar

Remarks

When you're using a scroll bar as an indicator of quantity or speed or as an input device, use the **Max** and **Min** properties to set the appropriate range for the control.

To specify the amount of change to report in a scroll bar, use the **LargeChange** property for clicking in the scroll bar, and the **SmallChange** property for clicking the arrows at the ends of the scroll bar. The scroll bar's **Value** property increases or decreases by the values set for the **LargeChange** and **SmallChange** properties. You can position the scroll box at run time by setting **Value** between 0 and 32,767, inclusive.

Image Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjImageC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjImageX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjImageP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjImageM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjImageE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjImageS"}

Use the **Image** control to display a graphic. An **Image** control can display a graphic from a bitmap, icon, or metafile, as well as enhanced metafile, JPEG, or GIF files.

Syntax

Image

Remarks

The **Image** control uses fewer system resources and repaints faster than a **PictureBox** control, but it supports only a subset of the **PictureBox** properties, events, and methods. Use the **Stretch** property to determine whether the graphic is scaled to fit the control or vice versa. Although you can place an **Image** control within a container, an **Image** control can't act as a container.

Label Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjLabelC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjLabelX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjLabelP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjLabelM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjLabelE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjLabelS"}

A **Label** control is a graphical control you can use to display text that a user can't change directly.

Syntax

Label

Remarks

You can write code that changes the text displayed by a **Label** control in response to events at run time. For example, if your application takes a few minutes to commit a change, you can display a processing-status message in a **Label**. You can also use a **Label** to identify a control, such as a **TextBox** control, that doesn't have its own **Caption** property.

Set the **AutoSize** and **WordWrap** properties if you want the **Label** to properly display variable-length lines or varying numbers of lines.

A **Label** control can also act as a destination in a DDEconversation. Set the **LinkTopic** property to establish a link, set the **LinkItem** property to specify an item for the conversation, and set the **LinkMode** property to activate the link. When these properties have been set, Visual Basic attempts to initiate the conversation and displays a message if it's unable to do so.

Set the **UseMnemonic** property to **True** if you want to define a character in the **Caption** property of the **Label** as an access key. When you define an access key in a **Label** control, the user can press and hold down ALT+ the character you designate to move the focus to the next control in the tab order.

Line Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjLineC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjLineX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjLineP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjLineM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjLineE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjLineS"}

A **Line** control is a graphical control displayed as a horizontal, vertical, or diagonal line.

Syntax

Line

Remarks

You can use a **Line** control at design time to draw lines on forms. At run time, you can use a **Line** control instead of, or in addition to, the **Line** method. Lines drawn with the **Line** control remain on the form even if the **AutoRedraw** property setting is **False**. **Line** controls can be displayed on forms, in picture boxes, and in frames. You can't use the **Move** method to move a **Line** control at run time, but you can move or resize it by altering its **X1**, **X2**, **Y1**, and **Y2** properties. The effect of setting the **BorderStyle** property depends on the setting of the **BorderWidth** property. If **BorderWidth** isn't 1 and **BorderStyle** isn't 0 or 6, **BorderStyle** is set to 1.

ListBox Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjListBoxC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjListBoxX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjListBoxP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjListBoxM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjListBoxE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjListBoxS"}

A **ListBox** control displays a list of items from which the user can select one or more. If the number of items exceeds the number that can be displayed, a scroll bar is automatically added to the **ListBox** control.

If no item is selected, the **ListIndex** property value is -1. The first item in the list is **ListIndex** 0, and the value of the **ListCount** property is always one more than the largest **ListIndex** value.

Syntax

ListBox

Remarks

To add or delete items in a **ListBox** control, use the **AddItem** or **RemoveItem** method. Set the **List**, **ListCount**, and **ListIndex** properties to enable a user to access items in the **ListBox**. Alternatively, you can add items to the list by using the **List** property at design time.

MDIForm Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjMDIFormC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjMDIFormX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjMDIFormP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjMDIFormM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjMDIFormE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjMDIFormS"}

An MDI (multiple-document interface) form is a window that acts as the background of an application and is the container for forms that have their **MDIChild** property set to **True**.

Syntax

MDIForm

Remarks

You create an **MDIForm** object by choosing MDI Form from the Insert menu.

An application can have only one **MDIForm** object but many MDI child forms. If an MDI child form has menus, the child form's menu bar automatically replaces the **MDIForm** object's menu bar when the MDI child form is active. A minimized MDI child form is displayed as an icon within the **MDIForm**.

An **MDIForm** object can contain only **Menu** and **PictureBox** controls and custom controls that have an **Align** property. To place other controls on an **MDIForm**, you can draw a picture box on the form, and then draw other controls inside the picture box. You can use the **Print** method to display text in a picture box on an **MDIForm**, but you can't use this method to display text on the **MDIForm** itself.

An **MDIForm** object can't be modal.

MDI child forms are designed independently of the **MDIForm**, but are always contained within the **MDIForm** at run time.

You can access the collection of controls on an **MDIForm** using the **Controls** collection. For example, to hide all the controls on an MDIForm you can use code similar to the following:

```
For Each Control in MDIForm1.Controls
   Control.Visible = False
Next Control
```

The **Count** property of the **MDIForm** tells you the number of controls in the **Controls** collection.

Menu Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjMenuC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjMenuX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjMenuP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjMenuM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjMenuE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjMenuS"}

A **Menu** control displays a custom menu for your application. A menu can include commands, submenus, and separator bars. Each menu you create can have up to four levels of submenus.

Syntax

Menu

Remarks

To create a **Menu** control, use the Menu Editor. Enter the name of the **Menu** control in the Caption box. To create a separator bar, enter a single hyphen (-) in the Caption box. To display a check mark to the left of a menu item, select the Checked box.

While you can set some **Menu** control properties using the Menu Editor, all **Menu** control properties are displayed in the Properties window. To display the properties of a **Menu** control, select the menu name in the Objects list at the top of the Properties window.

When you create an MDI application, the menu bar on the MDI child form replaces the menu bar on the **MDIForm** object when the child form is active.

OptionButton Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjOptionButtonC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjOptionButtonX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjOptionButtonP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjOptionButtonM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjOptionButtonE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjOptionButtonS"}

An **OptionButton** control displays an option that can be turned on or off.

Syntax

OptionButton

Remarks

Usually, **OptionButton** controls are used in an option group to display options from which the user selects only one. You group **OptionButton** controls by drawing them inside a container such as a **Frame** control, a **PictureBox** control, or a form. To group **OptionButton** controls in a **Frame** or **PictureBox**, draw the **Frame** or **PictureBox** first, and then draw the **OptionButton** controls inside. All **OptionButton** controls within the same container act as a single group.

While **OptionButton** controls and **CheckBox** controls may appear to function similarly, there is an important difference: When a user selects an **OptionButton**, the other **OptionButton** controls in the same group are automatically unavailable. In contrast, any number of **CheckBox** controls can be selected.

PictureBox Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjPictureBoxC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjPictureBoxX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjPictureBoxP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjPictureBoxM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjPictureBoxE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjPictureBoxS"}

A **PictureBox** control can display a graphic from a bitmap, icon, or metafile, as well as enhanced metafile, JPEG, or GIF files. It clips the graphic if the control isn't large enough to display the entire image.

Syntax

PictureBox

Remarks

You can also use a **PictureBox** control to group **OptionButton** controls and to display output from graphics methods and text written with the **Print** method.

To make a **PictureBox** control automatically resize to display an entire graphic, set its **AutoSize** property to **True**.

To create animation or simulation, you can manipulate graphics properties and methods in code. Graphics properties and events are useful for run-time print operations, such as modifying the format of a screen form for printing.

A **PictureBox** control can also act as a destination link in a DDE conversation.

The PictureBox and **Data** controls are the only standard Visual Basic controls that you can place in the internal area of an MDI form. You can use it to group controls at the top or bottom of the internal area to create a toolbar or status bar.

Shape Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjShapeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjShapeX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjShapeP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjShapeM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjShapeE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjShapeS"}

The **Shape** control is a graphical control displayed as a rectangle, square, oval, circle, rounded rectangle, or rounded square.

Syntax

Shape

Remarks

Use **Shape** controls at design time instead of, or in addition to, invoking **Circle** and **Line** methods at run time. You can draw a **Shape** control in a container, but it can't act as a container. The effect of setting the **BorderStyle** property depends on the setting of the **BorderWidth** property. If **BorderWidth** isn't 1 and **BorderStyle** isn't 0 or 6, **BorderStyle** is set to 1.

TextBox Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjTextBoxC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjTextBoxX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjTextBoxP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjTextBoxM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjTextBoxE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjTextBoxS"}

A **TextBox** control, sometimes called an edit field or edit control, displays information entered at design time, entered by the user, or assigned to the control in code at run time.

Syntax

TextBox

Remarks

To display multiple lines of text in a **TextBox** control, set the **MultiLine** property to **True**. If a multiple-line **TextBox** doesn't have a horizontal scroll bar, text wraps automatically even when the **TextBox** is resized. To customize the scroll bar combination on a **TextBox**, set the **ScrollBars** property.

If you set the **MultiLine** property to **True**, you can use the **Alignment** property to set the alignment of text within the **TextBox**. The text is left-justified by default. If the **MultiLine** property is **False**, setting the **Alignment** property has no effect.

A **TextBox** control can also act as a destination link in a DDE conversation.

Timer Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjTimerC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjTimerX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjTimerP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjTimerM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjTimerE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjTimerS"}

A **Timer** control can execute code at regular intervals by causing a Timer event to occur.

Syntax

Timer

Remarks

The **Timer** control, invisible to the user, is useful for background processing.

You can't set the **Enabled** property of a **Timer** for a multiple selection of controls other than **Timer** controls.

There is no practical limit on the number of active timer controls you can have in Visual Basic 5.0 running under Windows 95 or Windows NT.

Forms Collection Example

This example fills a list box with the captions of all the currently loaded forms.

```
Private Sub Form_Activate ()
   Dim I ' Declare variable.
   ' Refill list (in case an instance was added or removed).
   lstForms.Clear ' Clear list box.
   For I = 0 To Forms.Count - 1
      lstForms.AddItem Forms(I).Caption
   Next I
End Sub
```

Controls Collection Example

This example enables all currently loaded controls on a form (except menus).

```
Sub EnableControlsOn (Frm As Form, State As Integer)
   Dim I ' Declare variable.
   For I = 0 To Frm.Controls.Count - 1
      If Not TypeOf Frm.Controls(I) Is Menu Then
         Frm.Controls(I).Enabled = State
      End If
   Next I
End Sub
```

OLE Container Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjOLEContainerC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjOLEContainerX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjOLEContainerP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjOLEContainerM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjOLEContainerE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjOLEContainerS"}

The **OLE** container control enables you to add insertable objects to the forms in your Visual Basic applications. With the **OLE** container control, you can:

- Create a placeholder in your application for an insertable object. At run time you can create the object that is displayed within the **OLE** container control or change an object you placed within the **OLE** container control at design time.
- Create a linked object in your application.
- Bind the **OLE** container control to a database using the **Data** control.

You either create the object at design time using the Insert Object dialog box (which contains the commands Insert Object, Paste Special, and so on), or at run time by setting the appropriate properties.

When you move an **OLE** container control on a form using the **ObjectMove** method, the **Height** and **Width** property values of the object may be slightly different after the move. This is because the parameters to the **ObjectMove** method are pixel values converted to the current form's scaling mode. The conversion from pixels to twips and back doesn't always result in identical values.

Using the OLE Container Control's Pop-up Menus

Each time you draw an **OLE** container control on a form, the Insert Object dialog box is displayed. Use this dialog box to create a linked or embedded object. If you choose Cancel, no object is created.

At design time, click the **OLE** container control with the right mouse button to display a pop-up menu. The commands displayed on this pop-up menu depend on the state of the **OLE** container control as shown in the following table:

Command	**Enabled in pop-up menu when**
Insert Object	Always enabled.
Paste Special	**Clipboard** object contains a valid object.
Delete Embedded Object	**OLE** container control contains an embedded object.
Delete Linked Object	**OLE** container control contains a linked object.
Create Link	**SourceDoc** property is set.
Create Embedded Object	**Class** or **SourceDoc** property is set.

An **OLE** container control can contain only one object at a time. You can create a linked or embedded object in several ways:

- Use the Insert Object or Paste Special dialog boxes (run time or design time).
- Set the **Class** property in the Properties window, click the **OLE** container control with the right mouse button, and then select the appropriate command (design time only).
- Use the appropriate method of the **OLE** container control.

Finding Class Names

You can get a list of the class names available to your application by selecting the **Class** property in the Properties window and clicking the Properties button.

Note The Insert Object dialog box doesn't display a list of class names. This dialog box displays

user-friendly names for each class of object, which are generally longer and more easily understood.

Class Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"proClassC;vbproBooksOnlineJumpTopic;vbproClassC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproClassX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"proClassA;vbproClassA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproClassS"}

Returns or sets the class name of an embedded object .

Syntax

object.**Class** [**=** *string*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	A string expression specifying the class name.

Remarks

A class name defines the type of an object. Applications that support ActiveX components fully qualify the class names of their objects using either of the following syntaxes:

application.objecttype.version
objecttype.version

The syntax for ActiveX component class names has the following parts:

Part	Description
application	The name of the application that supplies the object.
objecttype	The object's name as defined in the object library.
version	The version number of the object or application that supplies the object.

For example, Microsoft Excel version 5.0 supports a number of objects, including worksheets and charts. Their class names are **Excel.Sheet.5** and **Excel.Chart.5**. Microsoft WordArt version 2.0 supports a single object with the class name **MSWordArt.2**.

Note Some ActiveX component programming documentation refers to the class name syntax as a programmatic ID.

To view a list of class names available on your system, select the **OLE** container control, select the **Class** property in the Properties window, and then click the builder button.

Copying an object from the system Clipboard updates the control's **Class** property. For example, if you paste a Microsoft Excel chart from the system Clipboard into an **OLE** container control that previously contained a Microsoft Excel worksheet, its **Class** property setting changes from **Excel.Sheet.5** to **Excel.Chart.5**. You can paste an object from the system Clipboard at run time with the **Paste** method or the **PasteSpecialDlg** method.

ObjectVerbFlags Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproObjectVerbFlagsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproObjectVerbFlagsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproObjectVerbFlagsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproObjectVerbFlagsS"}

Returns the menu state (such as enabled or disabled, checked, and so on) for each verb in a given **ObjectVerbs** array.

Syntax

object.**ObjectVerbFlags(***number***)**

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression indicating the element in the array.

Return Values

The **ObjectVerbFlags** property returns the following values:

Constant	Value	Description
vbOLEFlagChecked	&H0008	The menu item is checked.
vbOLEFlagDisabled	&H0002	The menu item is disabled (but not dimmed).
vbOLEFlagEnabled	&H0000	The menu item is enabled.
vbOLEFlagGrayed	&H0001	The menu item is dimmed.
vbOLEFlagSeparator	&H0800	The menu item is a separator bar.

Note These constants are also listed in the Visual Basic objects and procedures library in the Object Browser.

Remarks

The first verb in the **ObjectVerbs** array is the default verb. The **ObjectVerbFlags** array contains information about the menu state (such as dimmed, checked, and so on) for each verb in the **ObjectVerbs** array.

When displaying a menu containing an object's verbs, check the value of this property to see how the item is set to be displayed.

SizeMode Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSizeModeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSizeModeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSizeModeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSizeModeS"}

Returns or sets a value specifying how the **OLE** container control is sized or how its image is displayed when it contains an object.

Syntax

object.**SizeMode** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer or constant specifying how the control is sized or how its image is displayed, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
vbOLESizeClip	0	(Default) Clip. The object is displayed in actual size. If the object is larger than the **OLE** container control, its image is clipped by the control's borders.
vbOLESizeStretch	1	Stretch. The object's image is sized to fill the **OLE** container control. The image may not maintain the original proportions of the object.
vbOLESizeAutoSize	2	Autosize. The **OLE** container control is resized to display the entire object.
vbOLESizeZoom	3	Zoom. The object is resized to fill the **OLE** container control as much as possible while still maintaining the original proportions of the object.

Remarks

When **SizeMode** is set to 2 (Autosize), the **OLE** container control is automatically resized when the display size of an object changes. When this occurs, the Resize event is invoked before the **OLE** container control is automatically resized. The *heightnew* and *widthnew* arguments in the Resize event procedure indicate the optimal size for displaying the object (this size is determined by the application that created the object). You can size the control by changing the values of the *heightnew* and *widthnew* arguments in the Resize event procedure.

Data Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDataC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDataX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDataA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDataS"}

Returns or sets a handle to a memory object or graphical device interface (GDI) object containing data in a specified format. Not available at design time.

Syntax

object.**Data** [**=** *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A Long integer specifying the handle.

Remarks

Set this property to send data to an application that created an object. Before using the **Data** property, set the **Format** property to specify the type of data contained in the memory object or GDI object.

You can get a list of acceptable formats for an object using the **ObjectAcceptFormats** and **ObjectGetFormats** properties.

Setting this property to 0 frees the memory associated with the handle.

Tip Automation provides an easier and more reliable solution for sending data and commands to and from an object. If an object supports Automation, you can access the object through the **Object** property or using the **CreateObject** and **GetObject** functions

DataText Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDataTextC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDataTextX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDataTextA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDataTextS"}

Returns a string from or sets a string for the specified object.

Syntax

object.**DataText** [**=** *string*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	A string expression specifying the string.

Remarks

To send a string to an object, first set the **Format** property to a format the object supports. Use the **ObjectGetFormats** and **ObjectAcceptFormats** properties to get a list of formats supported by an object.

When getting data from an object, the **DataText** property returns the string sent from the object, ending at the first null character.

The **DataText** string can be as large as available memory permits.

Tip Automation provides an easier and more reliable solution for sending data and commands to and from an object. If an object supports Automation, you can access the object through the **Object** property or using the **CreateObject** and **GetObject** functions.

FileNumber Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFileNumberC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFileNumberX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFileNumberA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFileNumberS"}

Returns or sets the file number to be used when saving or loading an object, or returns the last file number used. Not available at design time.

Note The **FileNumber** property is included for compatibility with the **Action** property in earlier versions. For current functionality, use the **SaveToFile** and **ReadFromFile** methods.

Syntax

object.**FileNumber** [**=** *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression specifying the file number.

Remarks

The file number must correspond to an open, binary file.

You can use this property to specify the number of the file to be opened with the **ReadFromFile** method or saved with the **SaveToFile** or **SaveToOle1File** methods.

HostName Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHostNameC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHostNameX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHostNameA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHostNameS"}

Returns or sets the user-readable host name of your Visual Basic application.

Syntax

object.**HostName** [**=** *name*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
name	A string expression specifying the host name.

Remarks

When editing an object, the **HostName** property setting may be displayed in the object's window title. However, some applications that provide objects don't display **HostName**.

lpOleObject Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLpOleObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLpOleObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLpOleObjectA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLpOleObjectS"}

Returns the address of the object.

Syntax

object.**lpOleObject**

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

Remarks

Many function calls in the ActiveX DLLs require the address of an object as an argument. Pass the value specified in the **lpOleObject** property when making API calls to the ActiveX DLLs. The value is 0 if no object is currently displayed. If a call is made to an API that makes a callback to the **OLE** container control, the result is unpredictable.

The address returned by this property is a pointer to the **lpOleObject** interface for the active object.

ObjectAcceptFormatsCount Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproObjectAcceptFormatsCountC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproObjectAcceptFormatsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproObjectAcceptFormatsCountA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproObjectAcceptFormatsCountS"}

Returns the number of formats that can be accepted by an object.

Syntax

object.**ObjectAcceptFormatsCount**

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

Remarks

Use this property to get the number of elements in the **ObjectAcceptFormats** property array.

DisplayType Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDisplayTypeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDisplayTypeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDisplayTypeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDisplayTypeS"}

Returns or sets a value indicating whether an object displays its contents or an icon.

Syntax

object.**DisplayType** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer or constant specifying whether an object displays its contents or an icon, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Descriptioni
vbOLEDisplayContent	0	(Default) Content. When the **OLE** container control contains an object, the object's data is displayed in the control.
vbOLEDisplayIcon	1	Icon. When the **OLE** container control contains an object, the object's icon is displayed in the control.

Remarks

This property determines the default setting of the Display As Icon check box in the Insert Object and Paste Special dialog boxes. When you display these dialog boxes either at run time (with the **InsertObjDlg** or **PasteSpecialDlg** methods) or design time, the Display As Icon check box is automatically selected if this property is set to 1 (Icon).

When creating an object at run time using the **CreateEmbed** or **CreateLink** methods, use the **DisplayType** property to determine if the object is displayed as an icon (set **DisplayType** = 1) or if the object's data is displayed in the control (set **DisplayType** = 0).

Once you create an object, you can't change its display type.

ObjectGetFormats Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproObjectGetFormatsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproObjectAcceptFormatsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproObjectGetFormatsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproObjectGetFormatsS"}

Returns the list of formats an object can provide.

Syntax

object.**ObjectGetFormats(***number***)**

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression indicating the element in the array.

Remarks

The list is a zero-based string array. Elements of the array can be used to set the **Format** property when getting data from an object using the **Data** and **DataText** properties.

ObjectGetFormatsCount Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproObjectGetFormatsCountC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproObjectAcceptFormatsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproObjectGetFormatsCountA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproObjectGetFormatsCountS"}

Returns the number of formats an object can provide.

Syntax

object.**ObjectGetFormatsCount**

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

Remarks

Use this property to determine the number of elements in the **ObjectGetFormats** property array.

OLEType Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproOLETypeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLETypeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLETypeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLETypeS"}

Returns the status of the object in an **OLE** container control.

Syntax

object.**OLEType**

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

Return Values

The **OLEType** property returns the following values:

Constant	Value	Description
vbOLELinked	0	Linked. The **OLE** container control contains a linked object. All the object's data is managed by the application that created it. When the object is saved using the **SaveToFile** method, only link information such as **SourceDoc**, **SourceItem**, and so on is saved in the specified file by your Visual Basic application.
vbOLEEmbedded	1	Embedded. The **OLE** container control contains an embedded object. All the object's data is managed within the Visual Basic application. When the object is saved using the **SaveToFile** method, all data associated with the object is saved in the specified file.
vbOLENone	3	None. The **OLE** container control doesn't contain an object.

Remarks

Use this property to determine if the **OLE** container control contains an object or to determine the type of object the **OLE** container control contains.

Use the **AppIsRunning** property to determine if the application that created the object is running.

When creating an object, use the **OLETypeAllowed** property to determine the type of object that can be created.

ObjectVerbs Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproObjectVerbsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproObjectAcceptFormatsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproObjectVerbsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproObjectVerbsS"}

Returns the list of verbs an object supports.

Syntax

object.**ObjectVerbs**(*number*)

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression indicating the element in the array.

Remarks

ObjectVerbs is a zero-based string array. Use this property along with the **ObjectVerbsCount** property to get the verbs supported by an object. These verbs are used to determine an action to perform when an object is activated with the **DoVerb** method. The list of verbs in the array varies from object to object and depends on the current conditions.

Each object can support its own set of verbs. The following values represent standard verbs supported by every object:

Constant	Value	Description
vbOLEPrimary	0	The default action for the object.
vbOLEShow	-1	Activates the object for editing. If the application that created the object supports in-place activation, the object is activated within the **OLE** container control.
vbOLEOpen	-2	Opens the object in a separate application window. If the application that created the object supports in-place activation, the object is activated in its own window.
vbOLEHide	-3	For embedded objects, hides the application that created the object.
vbOLEUInPlaceUIActivate	-4	If the object supports in-place activation, activates the object for in-place activation and shows any user interface tools. If the object doesn't support in-place activation, the object doesn't activate, and an error occurs.
vbOLEInPlaceActivate	-5	If the user moves the focus to the **OLE** container control, creates a window for the object and prepares the object to be edited. An error occurs if the object doesn't support activation on a

		single mouse click.
vbOLEDiscardUndoState	-6	Used when the object is activated for editing to discard all record of changes that the object's application can undo.

Note These verbs may not be listed in the **ObjectVerbs** property array.

The first verb in the **ObjectVerbs** array, ObjectVerbs(0), is the default verb. Unless otherwise specified, this verb activates the object.

The remaining verbs in the array can be displayed on a menu. If it's appropriate to display the default verb in a menu, the default verb has two entries in the **ObjectVerbs** array.

Applications that display objects typically include an Object command on the Edit menu. When the user chooses Edit Object, a menu displays the object's verbs. Use the **ObjectVerbs**, **ObjectVerbsCount**, and **ObjectVerbFlags** properties to create such a menu at run time.

The list of verbs an object supports may vary, depending on the state of the object. To update the list of verbs an object supports, use the **FetchVerbs** method. Be sure to update the list of verbs before presenting it to the user.

To automatically display the verbs in the **ObjectVerbs** array in a pop-up menu when the user clicks an object with the right mouse button, set the **AutoVerbMenu** property to **True**.

ObjectVerbsCount Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproObjectVerbsCountC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproObjectAcceptFormatsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproObjectVerbsCountA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproObjectVerbsCountS"}

Returns the number of verbs supported by an object.

Syntax

object.**ObjectVerbsCount**

The *object* is an object expression that evaluates to an **OLE** container control.

Remarks

Use this property to determine the number of elements in the **ObjectVerbs** property array.

The list of verbs an object supports may vary, depending on the state of the object. To update the list of verbs an object supports, use the **FetchVerbs** method.

SourceDoc Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSourceDocC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSourceDocX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSourceDocA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSourceDocS"}

Returns or sets the filename to use when you create an object.

Note You set the **SourceDoc** property for compatibility with the **Action** property in earlier versions. For current functionality, use the **CreateEmbed** and **CreateLink** methods.

Syntax

object.**SourceDoc** [**=** *name*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
name	A string expression specifying a filename.

Remarks

Use the **SourceDoc** property to specify the file to be linked when creating a linked object using the **Action** property. Use the **SourceItem** property to specify data within the file to be linked.

When creating an embedded object using the **Action** property, if the **SourceDoc** property is set to a valid filename, an embedded object is created using the specified file as a template.

When a linked object is created, the **SourceItem** property is concatenated to the **SourceDoc** property. At run time, the **SourceItem** property returns a zero-length string (""), and the **SourceDoc** property returns the entire path to the linked file, followed by an exclamation point (**!**) or a backslash (\), followed by the **SourceItem**. For example:

"C:\WORK\QTR1\REVENUE.XLS!R1C1:R30C15"

SourceItem Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSourceItemC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSourceItemX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSourceItemA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSourceItemS"}

Returns or sets the data within the file to be linked when you create a linked object.

Syntax

object.**SourceItem** [**=** *string*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	A string expression specifying the data to be linked.

Remarks

OLETypeAllowed must be set to 0 (Linked) or 2 (Either) when using this property. Use the **SourceDoc** property to specify the file to link.

Each object uses its own syntax to describe units of data. To set this property, specify a unit of data recognized by the object. For example, when you link to Microsoft Excel, specify the **SourceItem** using a cell or cell-range reference such as R1C1 or R3C4:R9C22, or a named range such as Revenues.

To determine the syntax to describe a unit of data for an object, see the documentation for the application that created the object.

Note You may be able to determine this syntax by creating a linked object at design timeusing the Paste Special command (click the **OLE** container control with the right mouse button). Once the object is created, select the **SourceDoc** property in the Properties window and look at the string in the Settings box. For most objects, this string contains a path to the linked file, followed by an exclamation point (**!**) or a backslash (\) and the syntax for the linked data.

When a linked object is created, the **SourceItem** property is concatenated to the **SourceDoc** property. At run time, the **SourceItem** property returns a zero-length string (""), and the **SourceDoc** property returns the entire path to the linked file, followed by an exclamation point (**!**) or a backslash (\), followed by the **SourceItem**. For example:

```
"C:\WORK\QTR1\REVENUE.XLS!R1C1:R30C15"
```

AppIsRunning Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproAppIsRunningC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAppIsRunningX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproAppIsRunningA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAppIsRunningS"}

Returns or sets a value that indicates whether the application that created the object in the **OLE** container control is running. Not available at design time.

Syntax

object.**AppIsRunning** [**=** *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying whether or not the application that produced the object in the **OLE** container control is running, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The application that produced the object in the **OLE** container control is running.
False	The application that produced the object in the **OLE** container control isn't running.

Remarks

You can set the value of the **AppIsRunning** property to start the application that produces the object in the **OLE** container control. Doing this causes objects to activate more rapidly. You can also set this property to **False** to close the application when the object loses the focus.

UpdateOptions Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproUpdateOptionsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproUpdateOptionsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproUpdateOptionsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproUpdateOptionsS"}

Returns or sets a value specifying how an object is updated when linked data is modified.

Syntax

object.**UpdateOptions** [**=** *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A integer specifying how an object is updated, as described in Settings.

Settings

The settings for *number* are:

Constant	Value	Description
vbOLEAutomatic	0	(Default) Automatic. The object is updated each time the linked data changes.
vbOLEFrozen	1	Frozen. The object is updated whenever the user saves the linked data from within the application in which it was created.
vbOLEManual	2	Manual. The object is updated only by using the **Update** method.

Remarks

This property is useful for linked objects where other users or applications can access and modify the linked data.

When an object's data is changed, the Updated event is invoked.

Verb Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproVerbC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproVerbX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproVerbA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproVerbS"}

Returns or sets a value specifying an operation to perform when an object is activated using the **Action** property.

Note The **Verb** property is included for compatibility with the **Action** property in earlier versions. For current functionality, use the **DoVerb** method.

Syntax

object.**Verb** [**=** *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A value that specifies the operation to perform.

Remarks

Each object can support its own set of verbs. Use the **ObjectVerbs** and **ObjectVerbsCount** properties to access the list of verbs supported by an object. Set **Verb** = 1 to specify the first verb in the list, set **Verb** = 2 to specify the second verb in the list, and so on.

Set **AutoActivate** to 2 (Double-Click) to automatically activate an object when it's double-clicked by the user.

Set **AutoVerbMenu** = **True** to display a pop-up menu containing the object's verbs when the user clicks the object with the right mouse button.

Updated Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtUpdatedC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtUpdatedX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtUpdatedA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtUpdatedS"}

Occurs when an object's data has been modified.

Syntax

Sub *object*_**Updated (***code* **As Integer)**

The **Updated** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
code	An integer that specifies how the object was updated, as described in Settings.

Settings

The settings for *code* are:

Constant	Value	Description
vbOLEChanged	0	The object's data has changed.
vbOLESaved	1	The object's data has been saved by the application that created the object.
vbOLEClosed	2	The file containing the linked object's data has been closed by the application that created the object.
vbOLERenamed	3	The file containing the linked object's data has been renamed by the application that created the object.

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser. You can use this event to determine if an object's data has been changed since it was last saved. To do this, set a global variable in the Updated event indicating that the object needs to be saved. After you save the object, reset the variable.

AutoActivate Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAutoActivateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAutoActivateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAutoActivateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAutoActivateS"}

Returns or sets a value that enables the user to activate an object by double-clicking the **OLE** container control or by moving the focus to the **OLE** container control.

Syntax

object.**AutoActivate** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer or constant specifying the technique used to activate the object within the **OLE** container control, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
vbOLEActivateManual	0	Manual. The object isn't automatically activated. You can activate an object programmatically using the **DoVerb** method.
vbOLEActivateGetFocus	1	Focus. If the **OLE** container control contains an object that supports single click activation, the application that provides the object is activated when the **OLE** container control receives the focus.
vbOLEActivateDoubleclick	2	(Default) Double-Click. If the **OLE** container control contains an object, the application that provides the object is activated when the user double-clicks the **OLE** container control or presses ENTER when the control has the focus.
vbOLEActivateAuto	3	Automatic. If the **OLE** container control contains an object, the application that provides the object is activated based on the object's normal method of activation—either when the control receives the focus or when the user double-clicks the control.

Remarks

You can determine if the **OLE** container control contains an object by checking the **OLEType** property.

Note When **AutoActivate** is set to 2 (Double-Click), the DblClick event doesn't occur when the user double-clicks an **OLE** container control.

AutoVerbMenu Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAutoVerbMenuC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAutoVerbMenuX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAutoVerbMenuA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAutoVerbMenuS"}

Returns or sets a value that determines if a pop-up menu containing the object's verbs is displayed when the user clicks the **OLE** container control with the right mouse button.

Syntax

object.**AutoVerbMenu**[**=** *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying whether a pop-up menu is displayed, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) When the user clicks the **OLE** container control with the right mouse button, a pop-up menu is displayed, showing the commands the object supports.
False	No pop-up menu is displayed.

Remarks

When this property is set to **True**, Click events and MouseDown events don't occur when the **OLE** container control is clicked with the right mouse button.

In order to display your own menus, the **AutoVerbMenu** property must be set to **False**.

Object Property (OLE Container)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproObjectA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":""}

Returns the object and/or a setting of an object's method or property in an **OLE** container control.

Syntax

object.**Object**[.*property* | .*method*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
property	Property that the object supports.
method	Method that the object supports.

Remarks

Use this property to specify an object you want to use in an Automation task.

You use the object returned by the **Object** property in an Automation task by using the properties and methods of that object. For information on which properties and methods an object supports, see the documentation for the application that created the object.

OLETypeAllowed Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproOLETypeAllowedC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLETypeAllowedX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLETypeAllowedA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLETypeAllowedS"}

Returns or sets the type of object the **OLE** container control can contain.

Syntax

object.**OLETypeAllowed** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer or constant that specifies the type of object, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
vbOLELinked	0	Linked. The **OLE** container control can contain only a linked object.
vbOLEEmbedded	1	Embedded. The **OLE** container control can contain only an embedded object.
vbOLEEither	2	(Default) Either. The **OLE** container control can contain either a linked or an embedded object.

Remarks

This property determines the type of object a user can create:

- When using the Insert Object dialog box, use the **InsertObjDlg** method to display this dialog box.
- When using the Paste Special dialog box, use the **PasteSpecialDlg** method to display this dialog box.
- When pasting an object from the system Clipboard, use the **Paste** method to paste objects from the system Clipboard.

Use the **OLEType** property to determine an object's type (linked, embedded, or none).

PasteSpecialDlg Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthPasteSpecialDlgC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthPasteSpecialDlgX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthPasteSpecialDlgA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthPasteSpecialDlgS"}

Displays the Paste Special dialog box.

Syntax

object.**PasteSpecialDlg**

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

Remarks

At run time, you display this dialog box to enable the user to paste an object from the system Clipboard. This dialog box displays several options to the user, including pasting either a linked or embedded object.

Use the **OLETypeAllowed** property to determine the type of object that can be created (linked, embedded, or either) using this dialog box.

If the **PasteOK** property setting is **True** and Visual Basic can't paste the object, the **OLE** container control deletes any object already in the control.

Copy Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthCopyC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthCopyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthCopyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthCopyS"}

Copies the object within an **OLE** container control to the system Clipboard.

Syntax

object.**Copy**

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

Remarks

When you copy an object onto the system Clipboard, all the data and link information associated with the object is placed on the system Clipboard. You can copy both linked and embedded objects onto the system Clipboard.

You can use this method to support an Edit Copy command on a menu.

CreateEmbed Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthCreateEmbedC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthCreateEmbedX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthCreateEmbedA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthCreateEmbedS"}

Creates an embedded object. Doesn't support named arguments.

Syntax

object.**CreateEmbed** *sourcedoc*, *class*

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
sourcedoc	Required. The filename of a document used as a template for the embedded object. Must be a zero-length string ("") if you don't specify a source document.
class	Optional. The name of the class of the embedded object. Ignored if you specify a filename for *sourcedoc*.

Remarks

To view a list of valid class names available on your system, select a control, such as the **OLE** container control, select the **Class** property in the Properties window, and then click the builder button.

Note You don't need to set the **Class** and **SourceDoc** properties when using the **CreateEmbed** method to create an embedded object.

When you create a new object, the application associated with the class name (for example, Excel.exe) must be correctly registered with the operating system. (The application setup program should register the application correctly.)

CreateLink Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthCreateLinkC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthCreateLinkX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthCreateLinkA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthCreateLinkS"}

Creates a linked object from the contents of a file. Doesn't support named arguments.

Syntax

object.**CreateLink** *sourcedoc*, *sourceitem*

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
sourcedoc	Required. The file from which the object is created.
sourceitem	Optional. The data within the file to be linked in the linked object.

Remarks

If you specify values for the arguments of this method, those values override the settings of the **SourceDoc** and **SourceItem** properties. Those properties are updated to reflect the argument values when the method is invoked.

When an object is created with this method, the **OLE** container control displays an image of the file specified by the **SourceDoc** property. If the object is saved, only the link references are saved because the **OLE** container control contains only a metafile image of the data and no actual source data.

When you create a new object, the application associated with the class name (for example, Excel.exe) must be correctly registered with the operating system. (The application setup program should register the application correctly.)

Delete Method (OLE Container)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthDeleteC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthDeleteX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthDeleteA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthDeleteS"}

Deletes the specified object and frees the memory associated with it.

Syntax

object.**Delete**

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

Remarks

This method enables you to explicitly delete an object. Objects are automatically deleted when a form is closed or when the object is replaced with a new object.

InsertObjDlg Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthObjDialogC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthObjDialogX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthObjDialogA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthObjDialogS"}

Displays the Insert Object dialog box.

Syntax

object.**InsertObjDlg**

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

Remarks

At run time, you display this dialog box to enable the user to create a linked or embedded object by choosing the type of object (linked or embedded) and the application provides the object.

Use the **OLETypeAllowed** property to determine the type of object that can be created (linked, embedded, or either) using this dialog box.

When you create a new object, the application associated with the class name (for example, Excel.exe) must be correctly registered with the operating system. (The application setup program should register the application correctly.)

Paste Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthPasteC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthPasteX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthPasteA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthPasteS"}

Copies data from the system Clipboard to an **OLE** container control.

Syntax

object.**Paste**

The *object* is an object expression that evaluates to an object in the Applies To list

Remarks

To use this method, set the **OLETypeAllowed** property, and then check the value of the **PasteOK** property. You can't paste successfully unless **PasteOK** returns a value of **True**.

If the **Paste** method was carried out, the **OLEType** property is set to **vbOLELinked** (0) or **vbOLEEmbedded** (1). If the **Paste** method wasn't carried out, the **OLEType** property is set to **vbOLENone** (3).

You can use this method to support an Edit Paste command on a menu.

If the **PasteOK** property setting is **True** and Visual Basic can't paste the object, the **OLE** container control deletes any object already in the control.

ReadFromFile Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthReadFromFileC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthReadFromFileX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthReadFromFileA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthReadFromFileS"}

Loads an object from a data file created using the **SaveToFile** method. Doesn't support named arguments.

Syntax

object.**ReadFromFile** *filenumber*

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
filenumber	Required. A numeric expression specifying the file number used when loading an object. This number must correspond to an open, binary file.

Remarks

You can save an object to a data file using the **SaveToFile** or **SaveToOle1File** methods.

SaveToFile Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthSaveToFileC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthSaveToFileX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthSaveToFileA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthSaveToFileS"}

Saves an object to a data file. Doesn't support named arguments.

Syntax

object.**SaveToFile** *filenumber*

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
filenumber	Required. A numeric expression specifying the file number used when saving an object. This number must correspond to an open, binary file.

Remarks

Use this method to save ActiveX components. To save an ActiveX component in the OLE version 1.0 format, use the **SaveToOle1File** method instead.

If the object is linked (**OLEType** = **vbOLELinked**, 0), only the link information and an image of the data is saved to the specified file. The object's data is maintained by the application that created the object. If the object is embedded (**OLEType** = **vbOLEEmbedded**, 1), the object's data is maintained by the **OLE** container control and can be saved by your Visual Basic application.

You can load an object saved to a data file with the **ReadFromFile** method.

DoVerb Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthDoVerbC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthDoVerbX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthDoVerbA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthDoVerbS"}

Opens an object for an operation, such as editing. Doesn't support named arguments.

Syntax

object.**DoVerb** (*verb*)

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
verb	Optional. The verb to execute of the object within the **OLE** container control. If not specified, the default verb is executed. The value of this argument can be one of the standard verbs supported by all objects or an index of the **ObjectVerbs** property array.

Remarks

If you set the **AutoActivate** property to 2 (Double-Click), the **OLE** container control automatically activates the current object when the user double-clicks the control.

Each object can support its own set of verbs. The following values represent standard verbs every object should support:

Constant	Value	Description
vbOLEPrimary	0	The default action for the object.
vbOLEShow	-1	Activates the object for editing. If the application that created the object supports in-place activation, the object is activated within the **OLE** container control.
vbOLEOpen	-2	Opens the object in a separate application window. If the application that created the object supports in-place activation, the object is activated in its own window.
vbOLEHide	-3	For embedded objects, hides the application that created the object.
vbOLEUIActivate	-4	If the object supports in-place activation, activates the object for in-place activation and shows any user interface tools. If the object doesn't support in-place activation, the object doesn't activate, and an error occurs.
VbOLEInPlaceActivate	-5	If the user moves the focus to the **OLE** container control, creates a window for the object and prepares the object to be edited.

		An error occurs if the object doesn't support activation on a single mouse click.
VbOLEDiscardUndoState	-6	Used when the object is activated for editing to discard all record of changes that the object's application can undo.

Note These verbs may not be listed in the **ObjectVerbs** property array.

FetchVerbs Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthFetchVerbsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthFetchVerbsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthFetchVerbsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthFetchVerbsS"}

Updates the list of verbs an object supports.

Syntax

object.**FetchVerbs**

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

Remarks

You can read the updated list of verbs using the **ObjectVerbs** property.

Close Method (OLE Container)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthCloseC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthCloseX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthCloseA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthCloseS"}

Closes an object and terminates the connection to the application that provided the object.

Syntax

object.**Close**

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

Remarks

This method applies only to embedded objects and is equivalent to closing the object. It has no effect on linked objects.

Update Method (OLE Container)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthUpdateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthUpdateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthUpdateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthUpdateS"}

Retrieves the current data from the application that supplied the object and displays that data as a graphic in the **OLE** container control.

Syntax

object.**Update**

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

Action Property (OLE Container)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproActionOLEC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproActionOLEX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproActionOLEA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproActionOLES"}

Sets a value that determines an action. Not available at design time.

Note The **Action** property is included for compatibility with earlier versions. For current functionality, use the methods listed in Settings.

Syntax

object.**Action =** *value*

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A constant or integer specifying the type of action, as described in Settings.

Settings

The settings for *value* are:

Value	Description	Current method
0	Creates an embedded object.	**CreateEmbed**
1	Creates a linked object from the contents of a file.	**CreateLink**
4	Copies the object to the system Clipboard.	**Copy**
5	Copies data from the system Clipboard to an **OLE** container control.	**Paste**
6	Retrieves the current data from the application that supplied the object and displays that data as a picture in the **OLE** container control.	**Update**
7	Opens an object for an operation, such as editing.	**DoVerb**
9	Closes an object and terminates the connection to the application that provided the object.	**Close**
10	Deletes the specified object and frees the memory associated with it.	**Delete**
11	Saves an object to a data file.	**SaveToFile**
12	Loads an object that was saved to a data file.	**ReadFromFile**
14	Displays the Insert Object dialog box.	**InsertObjDlg**
15	Displays the Paste Special dialog box.	**PasteSpecialDlg**
17	Updates the list of verbs an object supports.	**FetchVerbs**
18	Saves an object to the OLE version 1.0 file format.	**SaveToOle1File**

MiscFlags Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMiscFlagsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMiscFlagsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies

To":"vbproMiscFlagsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMiscFlagsS"}

Returns or sets a value that determines access to one or more additional features of the **OLE** container control.

Syntax

object.**MiscFlags** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer or constant specifying access to an additional feature, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
vbOLEMiscFlagMemStorage	1	Causes the control to use memory to store the object while it's loaded.
VbOLEMiscFlagDisableInPlace	2	Overrides the control's default behavior of allowing in-place activation for objects that support it.

Remarks

The **vbOLEMiscFlagMemStorage** flag setting is faster than the object's default action, which is to store it on disk as a temporary file. This setting can, however, use a great deal of memory for objects whose data requires a lot of space, such as a bitmap for a paint program.

If an object supports in-place activation, you can use the **vbOLEMiscFlagDisableInPlace** setting to force the object to activate in a separate window.

To combine values, use the **Or** operator. For example, to combine both flags, you could use this code:

```
Ole1.MiscFlags = vbOLEMiscFlagMemStorage Or _ vbOLEMiscFlagDisableInPlace
```

SaveToOle1File Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthSaveToOle1C;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthSaveToOle1X":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthSaveToOle1A"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthSaveToOle1S"}

Saves an object in the OLE version 1.0 file format. Doesn't support named arguments.

Syntax

object.**SaveToOle1File** *filenumber*

The **SaveToOle1File** method syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
filenumber	Required. A numeric expression specifying the file number used when saving or loading an object. This number must correspond to an open, binary file.

Remarks

If the object is linked (**OLEType** = **vbOLELinked**, 0), only the link information and an image of the data is saved to the specified file. The object's data is maintained by the application that created the object. If the object is embedded (**OLEType** = **vbOLEEmbedded**, 1), the object's data is maintained by the **OLE** container control and can be saved by your Visual Basic application.

If you want to save the object in the current ActiveX component format, use the **SaveToFile** method instead.

Format Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFormatC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFormatX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFormatA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFormatS"}

Returns or sets the format when sending data to and getting data from an application that created an object. Not available at design time.

Syntax

object.**Format** [= *format*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
format	A string expression specifying the format used with the **Data** and **DataText** properties.

Remarks

Use the **ObjectAcceptFormats**, **ObjectAcceptFormatsCount**, **ObjectGetFormats**, and **ObjectGetFormatsCount** properties to get a list of the acceptable data formats for a specific class of object.

Many applications that provide objects support only one or two formats. For example, Microsoft Draw accepts only the CF_METAFILEPICT format. Although CF_METAFILEPICT resembles the intrinsic constant **vbCFMetafile** (numeric value 3) defined in the Visual Basic (VB) object library in the Object Browser, it's actually a string literal and is assigned as:

```
Ole1.Format = "CF_METAFILEPICT"
```

In many cases, the list of formats an object can accept (**ObjectAcceptFormats**) is different from the list of formats an object can provide (**ObjectGetFormats**).

PasteOK Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPasteOKC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPasteOKX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPasteOKA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPasteOKS"}

Returns a value that determines whether the contents of the system Clipboard can be pasted into the **OLE** container control.

Syntax

object.**PasteOK**

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

Remarks

When this property setting is **True**, you can paste the contents of the system Clipboard into the **OLE** container control.

Use the **OLETypeAllowed** property to specify the type of object (linked or embedded) you want to paste into the **OLE** container control. Once you successfully paste an object into the **OLE** container control, you can check the **OLEType** property setting to determine the type of object that was created.

You can use this property if your application supports a Paste command on an Edit menu. If **PasteOK** is **False**, disable the menu command; otherwise, it can be enabled. Enable and disable menu commands by setting their **Enabled** property to **True** or **False**, respectively.

You paste an object into the **OLE** container control with the **Paste** method.

To provide more flexibility to the user, display a Paste Special dialog box when the user chooses the Edit Paste command. (Set **OLETypeAllowed** = 2, and then use the **PasteSpecialDlg** method.) When this dialog box is displayed, an object is pasted onto the system Clipboard based on the user's selections in the dialog box.

DataText Property Example

This example sends data to the Microsoft Graph application, so you must have **MS Graph** installed on your system to run the example. (This is installed by most Microsoft Office components.) Create a form about one-half the size of the screen with a **CommandButton** control (Command1) in the upper-left corner of the form and an **OLE** container control (OLE1) placed below the **CommandButton**.

When you place the **OLE** container control on the form, the Insert Object dialog box is displayed. Choose Cancel and press F5 to run the example.

```
Private Sub Command1_Click ()
Dim Msg, NL, TB' Declare variables.
   TB = Chr(9) ' Tab character.
   NL = Chr(10)   ' Newline character.
   ' Create data to replace default Graph data.
   Msg = TB + "Drew" & TB & "Teresa" & TB & "Bob"
   Msg = Msg + NL & "Eric" & TB & "1" & TB & "2" & TB & "3"
   Msg = Msg + NL & "Ted" & TB & "11" & TB & "22" & TB & "33"
   Msg = Msg + NL & "Arthur" & TB & "21" & TB & "32" & TB & "23"
   ' Send the data using the DataText property.
   ' Activate MSGRAPH as hidden.
Ole1.DoVerb - 3
   If Ole1.AppIsRunning Then
      Ole1.DataText = Msg
      ' Update the object.
      Ole1.Update
   Else
      MsgBox "Graph isn't active."
   End If
End Sub
Sub Form_Load ()
   Ole1.Format = "CF_TEXT" ' Set the file format to text.
   Ole1.SizeMode = 2 ' Autosize.
   Ole1.CreateEmbed "", "MSGRAPH"
End Sub
```

ObjectAcceptFormats, ObjectAcceptFormatsCount, ObjectGetFormats, ObjectGetFormatsCount, ObjectVerbs, ObjectVerbsCount Properties Example

To run this example, place an **OLE** container control and three **ListBox** controls on a form. Paste the example code into the Declarations section of the form and press F5. When the Insert Object dialog box is displayed, select an application in the New Object list box and choose OK to create an object.

```
Private Sub Form_Click ()
   Dim I ' Declare variable.
   ' Display the Insert Object dialog box.
   Ole1.InsertObjDlg
   ' Update the list of available verbs.
   Ole1.FetchVerbs   ' Fetch verbs.
   ' Clear the list boxes.
   List1.Clear
   List2.Clear
   List3.Clear
   ' Fill the verbs list box. Because ObjectVerbs(0) is
   ' the default verb and is repeated in the ObjectVerbs()
   ' array, start the count at 1.
   For I = 1 To Ole1.ObjectVerbsCount - 1
      List1.AddItem Ole1.ObjectVerbs(I)
   Next I
   'Fill the Accept Formats list box.
   For I = 0 To Ole1.ObjectAcceptFormatsCount - 1
      List2.AddItem Ole1.ObjectAcceptFormats(I)
   Next I
   ' Fill the Get Formats list box.
   For I = 0 To Ole1.ObjectGetFormatsCount - 1
      List3.AddItem Ole1.ObjectGetFormats(I)
   Next I
End Sub
```

PasteOK Property Example

This example pastes an object in the **OLE** container control if the **PasteOK** property setting is **True**. Otherwise, the example displays a message box.

```
Private Sub mnuEditPaste_Click ()
   ' Check value of PasteOK.
   If Ole1.PasteOK Then
      Ole1.Paste        ' Enable Paste command if True.
   Else         ' Otherwise, disable Paste
      mnuEditPaste.Enabled = False ' menu command and give
      MsgBox "Can't paste."  ' appropriate message.
   End If
End Sub
```

ObjectMove Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtObjectMoveC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtObjectMoveX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtObjectMoveA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtObjectMoveS"}

Occurs immediately after the object within an **OLE** container control is moved or resized while the object is active.

Syntax

Private Sub *object*_**ObjectMove(***left* **As Single**, *top* **As Single**, *width* **As Single**, *height* **As Single)**

The ObjectMove event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
left	The coordinate of the left edge of the **OLE** container control immediately after it's moved or resized.
top	The coordinate of the top edge of the **OLE** container control immediately after it's moved or resized.
width	The width of the **OLE** container control immediately after it's moved or resized.
height	The height of the **OLE** container control immediately after it's moved or resized.

Remarks

When a user moves or resizes an **OLE** container control, your application can use the ObjectMove event to determine whether to actually change the size and position of the control. If the ObjectMove event procedure doesn't change the **OLE** container control's position or size, the object within the **OLE** container control returns to its original position and is informed of its new size. The coordinates passed as arguments to this event include the border of the **OLE** container control.

The ObjectMove and Resize events are triggered when the **OLE** container control receives information about the size of the object it contains. However, the Resize event doesn't receive any information about the position of the control. If the **OLE** container control is moved off the form, the arguments have negative or positive values that represent the position of the object relative to the top and left of the form.

OLEDropAllowed Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproOLEDropAllowedC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEDropAllowedX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEDropAllowedA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEDropAllowedS"}

Returns or sets a value that determines whether an **OLE** container control can be a drop target for OLE drag-and-drop operations.

Syntax

object.**OLEDropAllowed** [**=** *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying whether the **OLE** container control can be a drop target, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	When dragging an object that can be linked object or embedded object, a drop icon appears when the mouse pointer moves over the **OLE** container control. Dropping the object on the **OLE** container control has the same effect as pasting the object from the system Clipboard using the **Paste** method.
False	(Default) No drop icon appears over the **OLE** container control when dragging an object that can be linked or embedded. Dropping the object on the **OLE** container control has no effect on the control.

Remarks

The **MousePointer** property determines the shape of the mouse pointer when the **OLEDropAllowed** property is set to **True**. If the setting of the **MousePointer** property is 0 (Default), Visual Basic displays the standard drag-and-drop icon for the action taking place.

The setting of the **OLETypeAllowed** property must be 1 (**vbOLEEmbedded**) or 2 (**vbOLEEither**) to move or copy the object that can be linked or embedded, or 0 (**vbOLELinked**) or 2 to link the object. Dropping an object when **OLEDropAllowed** is set to **True** has the same effect on the **Class**, **SourceDoc**, and **SourceItem** property settings as using the **Paste** method of the **OLE** container control.

If the **OLEDropAllowed** property is set to **True**, the **OLE** container control doesn't receive DragDrop or DragOver events when dragging an object. Also, the setting of the **DragMode** property has no effect on the drag-and-drop behavior of the **OLE** container control when the **OLEDropAllowed** property is set to **True**.

Instancing Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproCreatableC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCreatableX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCreatableA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCreatableS"}

Sets a value that specifies whether you can create instances of a public class outside a project, and if so, how it will behave. Not available at run time.

Settings

The **Instancing** property has these settings:

Setting	Description
1	(Default) Private. Other applications aren't allowed access to type library information about the class, and cannot create instances of it. Private objects are only for use within your component.
2	PublicNotCreatable. Other applications can use objects of this class only if your component creates the objects first. Other applications cannot use the **CreateObject** function or the **New** operator to create objects from the class.
3	OnNewProcess. Allows other applications to create objects from the class, but every object of this class that a client creates starts a new instance of your component.
4	GlobalOnNewProcess. Similar to OnNewProcess, except that properties and methods of the class can be invoked as if they were simply global functions.
5	InSameProcess. Allows other applications to create objects from the class. One instance of your component can provide any number of objects created in this fashion, regardless of how many applications request them.
6	GlobalInSameProcess. Similar to InSameProcess, with one addition: properties and methods of the class can be invoked as if they were simply global functions. It's not necessary to explicitly create an instance of the class first, because one will automatically be created.

Setting	Applies to Project Type			
	ActiveX Exe	ActiveX DLL	ActiveX Contol	Std. Exe
Private	X	X	X	X
PublicNotCreatable	X	X	X	
OnNewProcess	X			
GlobalOnNewProcess	X			
InSameProcess	X	X		
GlobalInSameProcess	X	X		

Remarks

The **Instancing** property has been expanded in Visual Basic 5.0 to incorporate the functionality of the Visual Basic 4.0 **Public** property.

When a class is creatable, you can use any of the following techniques to create instances of the class from other applications:

- Use the **CreateObject** function, as in:

  ```
  Set MyInstance = CreateObject("MyProject.MyClass")
  ```

- Use the **Dim** statement within the same project (or outside the project if the **Public** property is also set to **True**), as in:

  ```
  Dim MyInstance As New MyClass
  ```

  The **New** keyword indicates that MyInstance is to be declared as a new instance of MyClass.

If the **Public** property is **False**, the setting of the **Instancing** property is ignored. You can always create instances of the class within the project that defines the class. If the **Public** property is **True**, the class is visible and therefore can be controlled by other applications once an instance of the class exists.

Initialize Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtInitializeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtInitializeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtInitializeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtInitializeS"}

Occurs when an application creates an instance of a **Form**, **MDIForm**, **User** control, **Property Page**, or class.

Syntax

Private Sub *object*_**Initialize( )**

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

Remarks

You trigger the Initialize event when you:

- Use the **CreateObject** function to create an instance of a class. For example:
  ```
  Set X = CreateObject("Project1.MyClass")
  ```
- Refer to a property or event of an automatically created instance of a form or class in your code. For example:
  ```
  MyForm.Caption = "Example"
  ```

Use this event to initialize any data used by the instance of the **Form**, **MDIForm**, or class. For a **Form** or **MDIForm**, the Initialize event occurs before the Load event.

Terminate Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtTerminateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtTerminateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtTerminateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtTerminateS"}

Occurs when all references to an instance of a **Form**, **MDIForm**, **User** control, **Property Page**, or class are removed from memory by setting all the variables that refer to the object to **Nothing** or when the last reference to the object falls out of scope.

Syntax

Private Sub *object*_**Terminate( )**

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

Remarks

For all objects except classes, the Terminate event occurs after the Unload event.

The Terminate event isn't triggered if the instances of the form or class were removed from memory because the application terminated abnormally. For example, if your application invokes the **End** statement before removing all existing instances of the class or form from memory, the Terminate event isn't triggered for that class or form.

Negotiate Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproNegotiateC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproNegotiateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproNegotiateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproNegotiateS"}

Sets a value that determines whether a control that can be aligned is displayed when an active object on the form displays one or more toolbars. Not available at run time.

Settings

The **Negotiate** property has these settings:

Setting	Description
True	If the control is aligned within the form (the **Align** property is set to a nonzero value), the control remains visible when an active object on the form displays a toolbar.
False	(Default) The control isn't displayed when an active object on the form displays a toolbar. The toolbar of the active object is displayed in place of the control.

Remarks

The **Negotiate** property exists for all controls with an **Align** property. You use the **Align** property to align the control within a **Form** or **MDIForm** object; however, the toolbar negotiation occurs only on the **MDIForm**. The aligned control must be on the **MDIForm**.

If the **NegotiateToolbars** property is set to **False**, the setting of the **Negotiate** property has no effect.

NegotiateMenus Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproNegotiateMenusC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproNegotiateMenusX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproNegotiateMenusA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproNegotiateMenusS"}

Sets a value that determines whether or not a form incorporates the menus from an object on the form on the form's menu bar. Not available at run time.

Settings

The **NegotiateMenus** property has these settings:

Setting	Description
True	(Default) When an object on the form is active for editing, the menus of that object are displayed on the form's menu bar.
False	Menus of objects on the form aren't displayed on the form's menu bar.

Remarks

Using the **NegotiateMenus** property, you determine if the menu bar of a form will share (or negotiate) space with the menus of an active object on the form. If you don't want to include the menus of the active object on the menu bar of your form, set **NegotiateMenus** to **False**.

You can't negotiate menus between an **MDIForm** object and an object on the **MDIForm**.

If **NegotiateMenus** is set to **True**, the form must have a menu bar defined, even if the menu bar isn't visible. If the **MDIChild** property of the form is set to **True**, the menus of the active object are displayed on the menu bar of the MDI parent window (**MDIForm** object).

When **NegotiateMenus** is set to **True**, you can use the **NegotiatePosition** property of individual **Menu** controls to determine the menus that your form displays along with the menus of the active object.

NegotiatePosition Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproNegotiatePositionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproNegotiatePositionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproNegotiatePositionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproNegotiatePositionS"}

Sets a value that determines whether or not top-level **Menu** controls are displayed on the menu bar while a linked object or embedded object object on a form is active and displaying its menus. Not available at run time.

Settings

The **NegotiatePosition** property has these settings:

Setting	Description
0	(Default) None. The menu isn't displayed on the menu bar when the object is active.
1	Left. The menu is displayed at the left end of the menu bar when the object is active.
2	Middle. The menu is displayed in the middle of the menu bar when the object is active.
3	Right. The menu is displayed at the right end of the menu bar when the object is active.

Remarks

Using the **NegotiatePosition** property, you determine the individual menus on the menu bar of your form that share (or negotiate) menu bar space with the menus of an active object on the form. Any menu with **NegotiatePosition** set to a nonzero value is displayed on the menu bar of the form along with menus from the active object.

If the **NegotiateMenus** property is set to **False**, the setting of the **NegotiatePosition** property has no effect.

NegotiateToolbars Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproNegotiateToolbarsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproNegotiateToolbarsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproNegotiateToolbarsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproNegotiateToolbarsS"}

Sets a value that determines whether the toolbars of an object on an MDI child form are displayed on the **MDIForm** when the object on the MDI child form is active. Not available at run time.

Settings

The **NegotiateToolbars** property has these settings:

Setting	Description
True	(Default) The **MDIForm** object displays the toolbars of the active object on the top or bottom of the **MDIForm**. The active object determines whether the toolbars are displayed at the top or bottom of the **MDIForm**.
False	The toolbars of the active object either aren't displayed at all or are displayed as floating tool palettes, as determined by the active object.

Remarks

Use the **NegotiateToolbars** property when creating a multiple-document interface (MDI) application that includes objects on MDI child forms. With this property, you determine how the active object displays its toolbars. By setting this property to **True**, the **MDIForm** shares (or negotiates) space at the top or bottom of the form to display the toolbars of the active object.

If the **MDIForm** also contains a toolbar, use the **Negotiate** property to determine how the various toolbars share the available space.

Clear Method (DataObject Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthClearMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthClearMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthClearMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthClearMethodS"}

Deletes the contents of the **DataObject** object.

Syntax

object.**Clear**

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

Remarks

This method is available only for component drag sources. If **Clear** is called from a component drop target, an error is generated.

Most components support manual OLE drag and drop events, and some support automatic OLE drag and drop events.

DataObject Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjDataObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjDataObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjDataObjectP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjDataObjectM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjDataObjectE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjDataObjectS"}

The **DataObject** object is a container for data being transferred from an component source to an component target. The data is stored in the format defined by the method using the **DataObject** object.

Syntax

DataObject

Remarks

The **DataObject**, which mirrors the **IDataObject** interface, allows OLE drag and drop and clipboard operations to be implemented.

Most components support manual OLE drag and drop events, and some support automatic OLE drag and drop events.

DataObjectFiles Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcolDataObjectFilesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbcolDataObjectFilesX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vamthItem;vbcolDataObjectFilesP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vamthClear;vamthRemove;vbcolDataObjectFilesM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbcolDataObjectFilesE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcolDataObjectFilesS"}

A collection whose elements represent a list of all filenames used by a **DataObject** object (such as the names of files that a user drags to or from the Windows File Explorer.)

Syntax

object.**DataObjectFiles(***index***)**

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

Part	Description
object	An object expression that evaluates to a **DataObject** object.
index	An integer with a range from 0 to `DataObjectFiles.Count - 1`.

Remarks

Note This collection is used by the **Files** property only when the data in the **DataObject** object is in the **vbCFFiles** format.

The **DataObjectFiles** collection is used by the **Files** property to store filenames in a **DataObject** object. It includes the **Remove**, **Add**, and **Clear** methods which allow you to manipulate its contents.

Files Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproFilesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFilesX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFilesS"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFilesA"}

Returns a **DataObjectFiles** collection, which in turn contains a list of all filenames used by a **DataObject** object (such as the names of files that a user drags to or from the Windows File Explorer.)

Syntax

object.**Files(***index***)**

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

Part	Description
object	An object expression that evaluates to a **DataObject** object.
index	An integer which is an index to an array of filenames.

Remarks

The **Files** collection is filled with filenames only when the **DataObject** object contains data of type **vbCFFiles**. (The **DataObject** object can contain several different types of data.) You can iterate through the collection to retrieve the list of file names.

The **Files** collection can be filled to allow Visual Basic applications to act as a drag source for a list of files.

GetData Method (DataObject Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthGetDataMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthGetDataMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthGetDataMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthGetDataMethodS"}

Returns data from a **DataObject** object in the form of a variant.

Syntax

object.**GetData (***format***)**

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
format	A constant or value that specifies the data format, as described in Settings. Parentheses must enclose the constant or value. If *format* is 0 or omitted, **GetData** automatically uses the appropriate format.

Settings

The settings for *format* are:

Constant	Value	Description
vbCFText	1	Text (.txt files)
vbCFBitmap	2	Bitmap (.bmp files)
vbCFMetafile	3	metafile (.wmf files)
vbCFEMetafile	14	Enhanced metafile (.emf files)
vbCFDIB	8	Device-independent bitmap (DIB)
vbCFPalette	9	Color palette
vbCFFiles	15	List of files
vbCFRTF	-16639	Rich text format (.rtf files)

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

It's possible for the **GetData** and **SetData** methods to use data formats other than those listed in Settings, including user-defined formats registered with Windows via the `RegisterClipboardFormat()` API function. However, there are a few caveats:

- The **SetData** method requires the data to be in the form of a byte array when it does not recognize the data format specified.
- The **GetData** method always returns data in a byte array when it is in a format that it doesn't recognize, although Visual Basic can transparently convert this returned byte array into other data types, such as strings.
- The byte array returned by **GetData** will be larger than the actual data when running on some operating systems, with arbitrary bytes at the end of the array. The reason for this is that Visual Basic does not know the data's format, and knows only the amount of memory that the operating system has allocated for the data. This allocation of memory is often larger than is actually required for the data. Therefore, there may be extraneous bytes near the end of the allocated memory segment. As a result, you must use appropriate functions to interpret the returned data in a meaningful way (such as truncating a string at a particular length with the **Left** function if the data is in a text format).

Note Not all applications support **vbcfBitmap** or **vbCFPalette**, so it is recommended that you use **vbCFDIB** whenever possible.

GetFormat Method (DataObject Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthGetFormatMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthGetFormatMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthGetFormatMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthGetFormatMethodS"}

Returns an boolean value indicating whether an item in the **DataObject** object matches a specified format. Doesn't support named arguments.

Syntax

object.**GetFormat** *format*

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
format	A constant or value that specifies the data format, as described in Settings.

Settings

The settings for *format* are:

Constant	Value	Description
vbCFText	1	Text (.txt files)
vbCFBitmap	2	Bitmap (.bmp files)
vbCFMetafile	3	metafile (.wmf files)
vbCFEMetafile	14	Enhanced metafile (.emf files)
vbCFDIB	8	Device-independent bitmap (DIB)
vbCFPalette	9	Color palette
vbCFFiles	15	List of files
vbCFRTF	-16639	Rich text format (.rtf files)

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

The **GetFormat** method returns **True** if an item in the **DataObject** object matches the specified format. Otherwise, it returns **False**.

OLECompleteDrag Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLECompleteDragEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLECompleteDragEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLECompleteDragEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLECompleteDragEventS"}

Occurs when a source component is dropped onto a target component, informing the source component that a drag action was either performed or canceled.

Syntax

Private Sub *object*_**CompleteDrag(**[*effect* **As Long**]**)**

The CompleteDrag event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
effect	A long integer set by the source object identifying the action that has been performed, thus allowing the source to take appropriate action if the component was moved (such as the source deleting data if it is moved from one component to another). The possible values are listed in Settings.

Settings

The settings for *effect* are:

Constant	Value	Description
vbDropEffectNone	0	Drop target cannot accept the data, or the drop operation was cancelled.
vbDropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
vbDropEffectMove	2	Drop results in a link to the original data being created between drag source and drop target.

Remarks

The OLECompleteDrag event is the final event to be called in an OLE drag/drop operation. This event informs the source component of the action that was performed when the object was dropped onto the target component. The target sets this value through the *effect* parameter of the OLEDragDrop event. Based on this, the source can then determine the appropriate action it needs to take. For example, if the object was moved into the target (**vbDropEffectMove**), the source needs to delete the object from itself after the move.

If **OLEDragMode** is set to **Automatic**, then Visual Basic handles the default behavior. The event still occurs, however, allowing the user to add to or change the behavior.

Most components support manual OLE drag and drop events, and some support automatic OLE drag and drop events.

OLEDrag Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthOLEDragMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthOLEDragMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthOLEDragMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthOLEDragMethodS"}

Causes a component to initiate an OLE drag/drop operation.

Syntax

object.**OLEDrag**

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

Remarks

When the **OLEDrag** method is called, the component's OLEStartDrag event occurs, allowing it to supply data to a target component.

OLEDragDrop Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLEDragDropEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLEDragDropEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLEDragDropEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLEDragDropEventS"}

Occurs when a source component is dropped onto a target component when the source component determines that a drop can occur.

Note This event occurs only if **OLEDropMode** is set to **1** (**Manual**).

Syntax

Private Sub *object*_**OLEDragDrop(***data* **As DataObject**, *effect* **As Long**, *button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**

The OLEDragDrop event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
data	A **DataObject** object containing formats that the source will provide and, in addition, possibly the data for those formats. If no data is contained in the **DataObject**, it is provided when the control calls the **GetData** method. The **SetData** and **Clear** methods cannot be used here.
effect	A long integer set by the target component identifying the action that has been performed (if any), thus allowing the source to take appropriate action if the component was moved (such as the source deleting the data). The possible values are listed in Settings.
button	An integer which acts as a bit field corresponding to the state of a mouse button when it is depressed. The left button is bit 0, the right button is bit 1, and the middle button is bit 2. These bits correspond to the values 1, 2, and 4, respectively. It indicates the state of the mouse buttons; some, all, or none of these three bits can be set, indicating that some, all, or none of the buttons are depressed.
shift	An integer which acts as a bit field corresponding to the state of the SHIFT, CTRL, and ALT keys when they are depressed. The SHIFT key is bit 0, the CTRL key is bit 1, and the ALT key is bit 2. These bits correspond to the values 1, 2, and 4, respectively. The *shift* parameter indicates the state of these keys; some, all, or none of the bits can be set, indicating that some, all, or none of the keys are depressed. For example, if both the CTRL and ALT keys were depressed, the value of *shift* would be 6.
x,y	A number which specifies the current location of the mouse pointer. The *x* and *y* values are always expressed in terms of the coordinate system set by the **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, and **ScaleTop** properties of the object.

Settings

The settings for *effect* are:

Constant	Value	Description
vbDropEffectNone	0	Drop target cannot accept the data.

vbDropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
vbDropEffectMove	2	Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

Remarks

The source ActiveX component should always mask values from the *effect* parameter to ensure compatibility with future implementations of ActiveX components. Presently, only three of the 32 bits in the *effect* parameter are used. In future versions of Visual Basic, however, these other bits may be used. Therefore, as a precaution against future problems, drag sources and drop targets should mask these values appropriately before performing any comparisons.

For example, a source component should not compare an *effect* against, say, **vbDropEffectCopy**, such as in this manner:

```
If Effect = vbDropEffectCopy...
```

Instead, the source component should mask for the value or values being sought, such as this:

```
If Effect And vbDropEffectCopy = vbDropEffectCopy...
```

-or-

```
If (Effect And vbDropEffectCopy)...
```

This allows for the definition of new drop effects in future versions of Visual Basic while preserving backwards compatibility with your existing code.

Most components support manual OLE drag and drop events, and some support automatic OLE drag and drop events.

OLEDragMode Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproOLEDragModePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEDragModePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEDragModePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEDragModePropertyS"}

Returns or sets whether the component or the programmer handles an OLE drag/drop operation.

Syntax

object.**OLEDragMode** = *mode*

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
mode	An integer which specifies the method with which an component handles OLE drag/drop operations, as described in Settings.

Settings

The settings for *mode* are:

Constant	Value	Description
vbOLEDragManual	0	(Default) Manual. The programmer handles all OLE drag/drop operations.
vbOLEDragAutomatic	1	Automatic. The component handles all OLE drag/drop operations.

Remarks

When **OLEDragMode** is set to **Manual**, you must call the **OLEDrag** method to start dragging, which then triggers the **OLEStartDrag** event.

When **OLEDragMode** is set to **Automatic**, the source component fills the **DataObject** object with the data it contains and sets the *effects* parameter before initiating the OLEStartDrag event (as well as the OLESetData and other source-level OLE drag/drop events) when the user attempts to drag out of the control. This gives you control over the drag/drop operation and allows you to intercede by adding other formats, or by overriding or disabling the automatic data and formats using the **Clear** or **SetData** methods.

If the source's **OLEDragMode** property is set to **Automatic**, and no data is loaded in the **OLEStartDrag** event, or *aftereffects* is set to **0**, then the OLE drag/drop operation does not occur.

Note If the **DragMode** property of a control is set to **Automatic**, the setting of **OLEDragMode** is ignored, because regular Visual Basic drag and drop events take precedence.

OLEDragOver Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLEDragOverEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLEDragOverEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLEDragOverEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLEDragOverEventS"}

Occurs when one component is dragged over another.

Syntax

Private Sub *object*_**OLEDragOver(***data* **As DataObject**, *effect* **As Long**, *button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single**, *state* **As Integer)**

The OLEDragOver event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
data	A **DataObject** object containing formats that the source will provide and, in addition, possibly the data for those formats. If no data is contained in the **DataObject**, it is provided when the control calls the **GetData** method. The **SetData** and **Clear** methods cannot be used here.
effect	A long integer initially set by the source object identifying all effects it supports. This parameter must be correctly set by the target component during this event. The value of *effect* is determined by logically **Or**'ing together all active effects (as listed in Settings). The target component should check these effects and other parameters to determine which actions are appropriate for it, and then set this parameter to one of the allowable effects (as specified by the source) to specify which actions will be performed if the user drops the selection on the component. The possible values are listed in Settings.
button	An integer which acts as a bit field corresponding to the state of a mouse button when it is depressed. The left button is bit 0, the right button is bit 1, and the middle button is bit 2. These bits correspond to the values 1, 2, and 4, respectively. It indicates the state of the mouse buttons; some, all, or none of these three bits can be set, indicating that some, all, or none of the buttons are depressed.
shift	An integer which acts as a bit field corresponding to the state of the SHIFT, CTRL, and ALT keys when they are depressed. The SHIFT key is bit 0, the CTRL key is bit 1, and the ALT key is bit 2. These bits correspond to the values 1, 2, and 4, respectively. The *shift* parameter indicates the state of these keys; some, all, or none of the bits can be set, indicating that some, all, or none of the keys are depressed. For example, if both the CTRL and ALT keys are depressed, the value of *shift* would be 6.
x,y	A number that specifies the current horizontal (*x*) and vertical (*y*) position of the mouse pointer within the target form or control. The *x* and *y* values are always expressed in terms of the coordinate system set by the **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, and **ScaleTop** properties of the object.
state	An integer that corresponds to the transition state of the control being dragged in relation to a target form or control.

The possible values are listed in Settings.

Settings

The settings for *effect* are:

Constant	Value	Description
vbDropEffectNone	0	Drop target cannot accept the data.
vbDropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
vbDropEffectMove	2	Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.
vbDropEffectScroll	-2147483648 (&H80000000)	Scrolling is occuring or about to occur in the target component. This value is used in conjunction with the other values. **Note** Use only if you are performing your own scrolling in the target component.

The settings for *state* are:

Constant	Value	Description
vbEnter	0	Source component is being dragged within the range of a target.
vbLeave	1	Source component is being dragged out of the range of a target.
vbOver	2	Source component has moved from one position in the target to another.

Remarks

Note If the *state* parameter is **vbLeave**, indicating that the mouse pointer has left the target, then the *x* and *y* parameters will contain zeros.

The source component should always mask values from the *effect* parameter to ensure compatibility with future implementations of ActiveX components. Presently, only three of the 32 bits in the *effect* parameter are used. In future versions of Visual Basic, however, these other bits may be used. Therefore, as a precaution against future problems, drag sources and drop targets should mask these values appropriately before performing any comparisons.

For example, a source component should not compare an *effect* against, say, **vbDropEffectCopy**, such as in this manner:

```
If Effect = vbDropEffectCopy...
```

Instead, the source component should mask for the value or values being sought, such as this:

```
If Effect And vbDropEffectCopy = vbDropEffectCopy...
```

-or-

```
If (Effect And vbDropEffectCopy)...
```

This allows for the definition of new drop effects in future versions of Visual Basic while preserving backwards compatibility with your existing code.

Most components support manual OLE drag and drop events, and some support automatic OLE drag and drop events.

OLEDropMode Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproOLEDropModePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEDropModePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEDropModePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEDropModePropertyS"}

Returns or sets how a target component handles drop operations.

Syntax

object.**OLEDropMode** [= *mode*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
mode	An enumerated integer which specifies the method which a component handles OLE drag/drop operations, as described in Settings.

Settings

The settings for *mode* are:

Constant	Value	Description
vbOLEDropNone	0	(Default) None. The target component does not accept OLE drops and displays the No Drop cursor.
vbOLEDropManual	1	Manual. The target component triggers the OLE drop events, allowing the programmer to handle the OLE drop operation in code.
vbOLEDropAutomatic	2	Automatic. The target component automatically accepts OLE drops if the **DataObject** object contains data in a format it recognizes. No mouse or OLE drag/drop events on the target will occur when **OLEDropMode** is set to **vbOLEDropAutomatic**.

Remarks

Note The target component inspects what is being dragged over it in order to determine which events to trigger; the OLE drag/drop events, or the Visual Basic drag/drop events. There is no collision of components or confusion about which events are fired, since only one type of object can be dragged at a time.

OLEGiveFeedback Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLEGiveFeedbackEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLEGiveFeedbackEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLEGiveFeedbackEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLEGiveFeedbackEventS"}

Occurs after every OLEDragOver event. OLEGiveFeedback allows the source component to provide visual feedback to the user, such as changing the mouse cursor to indicate what will happen if the user drops the object, or provide visual feedback on the selection (in the source component) to indicate what will happen.

Syntax

Private Sub *object*_**OLEGiveFeedback(***effect* **As Long**, defaultcursors **As Boolean)**

The OLEGiveFeedback event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
effect	A long integer set by the target component in the OLEDragOver event specifying the action to be performed if the user drops the selection on it. This allows the source to take the appropriate action (such as giving visual feedback). The possible values are listed in Settings.
defaultcursors	A boolean value which determines whether Visual Basic uses the default mouse cursor proved by the component, or uses a user-defined mouse cursor. **True** (default) = use default mouse cursor. **False** = do not use default cursor. Mouse cursor must be set with the **MousePointer** property of the **Screen** object.

Settings

The settings for *effect* are:

Constant	Value	Description
vbDropEffectNone	0	Drop target cannot accept the data.
vbDropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
vbDropEffectMove	2	Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.
vbDropEffectScroll	-2147483648 (&H80000000)	Scrolling is occuring or about to occur in the target component. This value is used in conjunction with the other values. **Note** Use only if you are performing your own scrolling in the target component.

Remarks

If there is no code in the OLEGiveFeedback event, or if the *defaultcursors* parameter is set to **True**, then Visual Basic automatically sets the mouse cursor to the default cursor provided by the component.

The source component should always mask values from the *effect* parameter to ensure compatibility with future implementations of components. Presently, only three of the 32 bits in the *effect* parameter are used. In future versions of Visual Basic, however, these other bits may be used. Therefore, as a precaution against future problems, drag sources and drop targets should mask these values appropriately before performing any comparisons.

For example, a source component should not compare an *effect* against, say, **vbDropEffectCopy**, such as in this manner:

```
If Effect = vbDropEffectCopy...
```

Instead, the source component should mask for the value or values being sought, such as this:

```
If Effect And vbDropEffectCopy = vbDropEffectCopy...
```

-or-

```
If (Effect And vbDropEffectCopy)...
```

This allows for the definition of new drop effects in future versions of Visual Basic while preserving backwards compatibility with your existing code.

Most components support manual OLE drag and drop events, and some support automatic OLE drag and drop events.

OLESetData Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLESetDataEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLESetDataEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLESetDataEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLESetDataEventS"}

Occurs on an source component when a target component performs the **GetData** method on the source's **DataObject** object, but the data for the specified format has not yet been loaded.

Syntax

Private Sub *object*_**OLESetData(***data* **As DataObject**, *dataformat* **As Integer)**

The OLESetData event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
data	A **DataObject** object in which to place the requested data. The component calls the **SetData** method to load the requested format.
dataformat	An integer specifying the format of the data that the target component is requesting. The source component uses this value to determine what to load into the **DataObject** object.

Remarks

In certain cases, you may wish to defer loading data into the **DataObject** object of a source component to save time, especially if the source component supports many formats. This event allows the source to respond to only one request for a given format of data. When this event is called, the source should check the *format* parameter to determine what needs to be loaded and then perform the **SetData** method on the **DataObject** object to load the data which is then passed back to the target component.

OLEStartDrag Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLEStartDragEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLEStartDragEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLEStartDragEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLEStartDragEventS"}

Occurs when a component's **OLEDrag** method is performed, or when a component initiates an OLE drag/drop operation when the **OLEDragMode** property is set to **Automatic**.

This event specifies the data formats and drop effects that the source component supports. It can also be used to insert data into the **DataObject** object.

Syntax

Private Sub *object*_**StartDrag(***data* **As DataObject**, *allowedeffects* **As Long)**

The StartDrag event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
data	A **DataObject** object containing formats that the source will provide and, optionally, the data for those formats. If no data is contained in the **DataObject**, it is provided when the control calls the **GetData** method. The programmer should provide the values for this parameter in this event. The **SetData** and **Clear** methods cannot be used here.
allowedeffects	A long integer containing the effects that the source component supports. The possible values are listed in Settings. The programmer should provide the values for this parameter in this event.

Settings

The settings for *allowedeffects* are:

Constant	Value	Description
vbDropEffectNone	0	Drop target cannot accept the data.
vbDropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
vbDropEffectMove	2	Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

Remarks

The source component should logically **Or** together the supported values and places the result in the *allowedeffects* parameter. The target component can use this value to determine the appropriate action (and what the appropriate user feedback should be).

The StartDrag event also occurs if the component's **OLEDragMode** property is set to **Automatic**. This allows you to add formats and data to the **DataObject** object after the component has done so. You can also override the default behavior of the component by clearing the **DataObject** object (using the **Clear** method) and then adding your data and formats.

You may wish to defer putting data into the **DataObject** object until the target component requests it. This allows the source component to save time by not loading multiple data formats. When the target

performs the **GetData** method on the **DataObject**, the source's OLESetData event will occur if the requested data is not contained in the **DataObject**. At this point, the data can be loaded into the **DataObject**, which will in turn provide the data to the target.

If the user does not load any formats into the **DataObject**, then the drag/drop operation is canceled.

SetData Method (DataObject Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthSetDataMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthSetDataMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthSetDataMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthSetDataMethodS"}

Inserts data into a **DataObject** object using the specified data format.

Syntax

object.**SetData [***data***]**, **[***format***]**

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
data	Optional A variant containing the data to be passed to the **DataObject** object.
format	Optional. A constant or value that specifies the format of the data being passed, as described in Settings.

Settings

The settings for *format* are:

Constant	Value	Description
vbCFText	1	Text (.txt files)
vbCFBitmap	2	Bitmap (.bmp files)
vbCFMetafile	3	Metafile (.wmf files)
vbCFEMetafile	14	Enhanced metafile (.emf files)
vbCFDIB	8	Device-independent bitmap (DIB)
vbCFPalette	9	Color palette
vbCFFiles	15	List of files
vbCFRTF	-16639	Rich text format (.rtf files)

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

The *data* argument is optional. This allows you to set several different formats that the source component can support without having to load the data separately for each format. Multiple formats are set by calling **SetData** several times, each time using a different format. If you wish to start fresh, use the **Clear** method to clear all data and format information from the **DataObject**.

The *format* argument is also optional, but either the *data* or *format* argument must be specified. If *data* is specified, but not *format*, then Visual Basic will try to determine the format of the data. If it is unsuccessful, then an error is generated. When the target requests the data, and a format was specified, but no data was provided, the source's **OLESetData** event occurs, and the source can then provide the requested data type.

It's possible for the **GetData** and **SetData** methods to use data formats other than those listed in Settings, including user-defined formats registered with Windows via the `RegisterClipboardFormat()` API function. However, there are a few caveats:

- The **SetData** method requires the data to be in the form of a byte array when it does not recognize the data format specified.
- The **GetData** method always returns data in a byte array when it is in a format that it doesn't

recognize, although Visual Basic can transparently convert this returned byte array into other data types, such as strings.

- The byte array returned by **GetData** will be larger than the actual data when running on some operating systems, with arbitrary bytes at the end of the array. The reason for this is that Visual Basic does not know the data's format, and knows only the amount of memory that the operating system has allocated for the data. This allocation of memory is often larger than is actually required for the data. Therefore, there may be extraneous bytes near the end of the allocated memory segment. As a result, you must use appropriate functions to interpret the returned data in a meaningful way (such as truncating a string at a particular length with the **Left** function if the data is in a text format).

Jump to Visual Basic 5.0 Books Online

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":""} {ewc HLP95EN.DLL,DYNALINK,"Specifics":""}

No VB user should ever see this topic in Help. Leave this topic here, however. Every See Also ALink in the entire VB5 Help system calls this topic to start Books Online.

Visual Basic 5.0 Books Online Not Found

The compact disk or directory where the Visual Basic 5.0 Books Online are located was not found.

If Visual Basic 5.0 Books Online viewer has not been installed, install it and try the See Also link again.

If the Visual Basic 5.0 CD is not in your CD drive, please insert it in the drive and then try the See Also link again.

If these do not work, make sure that the location of the VBOnline.exe file that was installed to your hard drive for Visual Basic 5.0 Books Online is included in the WINHELP.INI file.

BackColor, ForeColor Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBackColorC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBackColorX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBackColorA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBackColorS"}

- **BackColor** — returns or sets the background color of an object.
- **ForeColor** — returns or sets the foreground color used to display text and graphics in an object.

Syntax

object.**BackColor** [= *color*]
object.**ForeColor** [= *color*]

The **BackColor** and **ForeColor** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
color	A value or constant that determines the background or foreground colors of an object, 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
Normal RGB colors	Colors specified by using the Color palette or by using the **RGB** or **QBColor** functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.

For all forms and controls, the default settings at design time are:

- **BackColor** — set to the system default color specified by the constant **vbWindowBackground**.
- **ForeColor** — set to the system default color specified by the constant **vbWindowText**.

Remarks

In the **Label**, and **Shape**, controls, the **BackColor** property is ignored if the **BackStyle** property setting is 0 (Transparent).

If you set the **BackColor** property on a **Form** object or a **PictureBox** control, all text and graphics, including the persistent graphics, are erased. Setting the **ForeColor** property doesn't affect graphics or print output already drawn. On all other controls, the screen color changes immediately.

The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF). The high byte of a number in this range equals 0; the lower 3 bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF). If the high byte isn't 0, Visual Basic uses the system colors, as defined in the user's Control Panel settings and by constants listed in the Visual Basic (VB) object library in the Object Browser.

To display text in the Windows operating environment, both the text and background colors must be solid. If the text or background colors you've selected aren't displayed, one of the selected colors may

be dithered — that is, comprised of up to three different-colored pixels. If you choose a dithered color for either the text or background, the nearest solid color will be substituted.

BackStyle Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBackStyleC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBackStyleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBackStyleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBackStyleS"}

Returns or sets a value indicating whether a **Label** control or the background of a **Shape** control is transparent or opaque.

Syntax

object.**BackStyle** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression specifying transparency, as described in Settings.

Settings

The settings for *number* are:

Setting	Description
0	Transparent — background color and any graphics are visible behind the control.
1	(Default) Opaque — the control's **BackColor** property setting fills the control and obscures any color or graphics behind it.

Remarks

You can use the **BackStyle** property to create transparent controls when you're using a background color on a **Form** object or **PictureBox** control or when you want to place a control over a graphic. Use an opaque control when you want it to stand out.

A control's **BackColor** property is ignored if **BackStyle** = 0.

BorderColor Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproBorderColorC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBorderColorX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBorderColorA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBorderColorS"}

Returns or sets the color of an object's border.

Syntax

object.**BorderColor** [= *color*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
color	A value or constant that determines the border color, 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
Normal RGB colors	Colors specified using the Color palette or by using the **RGB** or **QBColor** functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser . The system default color is specified by the **vbWindowText** constant. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.

Remarks

The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF). The high byte of a number in this range equals 0; the lower 3 bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF). If the high byte isn't 0, Visual Basic uses the system colors, as defined in the user's Control Panel settings and by constants listed in the Visual Basic (VB) object library in the Object Browser.

BorderStyle Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproBorderStyleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBorderStyleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBorderStyleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBorderStyleS"}

Returns or sets the border style for an object. For the **Form** object and the **TextBox** control, read-only at run time.

Syntax

object.**BorderStyle** = [*value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A value or constant that determines the border style, as described in Settings.

Settings

The **BorderStyle** property settings for a **Form** object are:

Constant	Setting	Description
vbBSNone	0	None (no border or border-related elements).
VbFixedSingle	1	Fixed Single. Can include Control-menu box, title bar, Maximize button, and Minim Maximize and Minimize buttons.
VbSizable	2	(Default) Sizable. Resizable using any of the optional border elements listed for se
VbFixedDouble	3	Fixed Dialog. Can include Control-menu box and title bar; can't include Maximize resizable.
VbFixedToolWindow	4	Fixed ToolWindow. Under Windows 3.x and Windows NT 3.51 and earlier, behave display Maximize or Minimize buttons. Not resizable. Under Windows 95, displays the title bar text in a reduced font size. The form does not appear in the Windows
VbSizableToolWindow	5	Sizable ToolWindow. Under Windows 3.x and Windows NT 3.51 and earlier, beha Maximize or Minimize buttons. Resizable. Under Windows 95, displays the Close text in a reduced font size. The form does not appear in the Windows 95 task bar.

The **BorderStyle** property settings for **MS Flex Grid**, **Image**, **Label**, **OLE** container, **PictureBox**, **Frame**, and **TextBox** controls are:

Setting	Description
0	(Default for **Image** and **Label** controls) None.
1	(Default for **MS Flex Grid**, **PictureBox**, **TextBox**, and **OLE** container controls) Fixed Single.

The **BorderStyle** property settings for **Line** and **Shape** controls are:

Constant	Setting	Description
vbTransparent	0	Transparent
vbBSSolid	1	(Default) Solid. The border is centered on the edge of the shape.
vbBSDash	2	Dash
vbBSDot	3	Dot
vbBSDashDot	4	Dash-dot

vbBSDashDotDot	5	Dash-dot-dot
vbBSInsideSolid	6	Inside solid. The outer edge of the border is the outer edge of the shape.

Remarks

For a form, the **BorderStyle** property determines key characteristics that visually identify a form as either a general-purpose window or a dialog box. Setting 3 (Fixed Dialog) is useful for standard dialog boxes. Settings 4 (Fixed ToolWindow) and 5 (Sizable ToolWindow) are useful for creating toolbox-style windows.

MDI child forms set to 2 (Sizable) are displayed within the MDI form in a default size defined by the Windows operating environment at run time. For any other setting, the form is displayed in the size specified at design time.

Changing the setting of the **BorderStyle** property of a **Form** object may change the settings of the **MinButton**, **MaxButton**, and **ShowInTaskbar** properties. When **BorderStyle** is set to 1 (Fixed Single) or 2 (Sizable), the **MinButton**, **MaxButton**, and **ShowInTaskbar** properties are automatically set to **True**. When **BorderStyle** is set to 0 (None), 3 (Fixed Dialog), 4 (Fixed ToolWindow), or 5 (Sizable ToolWindow), the **MinButton**, **MaxButton**, and **ShowInTaskbar** properties are automatically set to **False**.

Note If a form with a menu is set to 3 (Fixed Dialog), it is displayed with a setting 1 (Fixed Single) border instead.

At run time, a form is either modal or modeless, which you specify using the **Show** method.

BorderWidth Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproBorderWidthC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBorderWidthX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBorderWidthA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBorderWidthS"}

Returns or sets the width of a control's border.

Syntax

object.**BorderWidth** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression from 1 to 8192, inclusive.

Remarks

Use the **BorderWidth** and **BorderStyle** properties to specify the kind of border you want for a **Line** or **Shape** control. The following table shows the effect of **BorderStyle** settings on the **BorderWidth** property:

BorderStyle	Effect on BorderWidth
0	**BorderWidth** setting is ignored.
1–5	The border width expands from the center of the border; the height and width of the control are measured from the center of the border.
6	The border width expands inward on the control from the outside of the border; the height and width of the control are measured from the outside of the border.

If the **BorderWidth** property setting is greater than 1, the only effective settings of **BorderStyle** are 1 (Solid) and 6 (Inside Solid).

Cancel Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproCancelC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCancelX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCancelA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCancelS"}

Returns or sets a value indicating whether a command button is the Cancel button on a form. This command button can be the **CommandButton** control or any object within an **OLE** container control that behaves as a command button.

Syntax

object.**Cancel** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying whether the object is the Cancel button, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The **CommandButton** control is the Cancel button.
False	(Default) The **CommandButton** control isn't the Cancel button.

Remarks

Use the **Cancel** property to give the user the option of canceling uncommitted changes and returning the form to its previous state.

Only one **CommandButton** control on a form can be the Cancel button. When the **Cancel** property is set to **True** for one **CommandButton**, it's automatically set to **False** for all other **CommandButton** controls on the form. When a **CommandButton** control's **Cancel** property setting is **True** and the form is the active form, the user can choose the **CommandButton** by clicking it, pressing the ESC key, or pressing ENTER when the button has the focus.

For **OLE** container controls, the **Cancel** property is provided only for those objects that specifically behave as command buttons.

Tip For a form that supports irreversible operations, such as deletions, it's a good idea to make the Cancel button the default button. To do this, set both the **Cancel** property and the **Default** property to **True**.

Col, Row Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproColC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproColX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproColA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCols"}

Return or set the active cell in a **DBGrid** control. Not available at design time.

Syntax

object.**Col** [= *number*]
object.**Row** [= *number*]

The **Col** and **Row** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	The number of the column or row containing the active cell.

Remarks

Use these properties to specify a cell in a **DBGrid** control or to find out which column or row contains the active cell in a selected region. Columns and rows are numbered from zero, beginning at the top for rows and at the left for columns. Setting these properties at run time doesn't change which cells are selected. Use the **SelEndCol**, **SelStartCol**, **SelEndRow**, and **SelStartRow** properties to specify a selected region.

Note The **Col** and **Row** properties aren't the same as the **Cols** and **Rows** properties.

Cols, Rows Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproColsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproColX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproColsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproColsS"}

Return or set the total number of columns or rows in a **DBGrid** control.

Syntax

object.**Cols** [= *number*]
object.**Rows** [= *number*]

The **Cols** and **Rows** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	The number of columns or rows in a **DBGrid** control. The minimum number of columns is 1, the maximum is 400; the minimum number of rows is 1, the maximum is 2000.

Remarks

Use these properties to expand a **DBGrid** control dynamically at run time. A **DBGrid** control must have at least one non-fixed column and one non-fixed row.

Note The **Cols** and **Rows** properties aren't the same as the **Col** and **Row** properties.

ControlBox Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproControlBoxC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproControlBoxX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproControlBoxA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproControlBoxS"}

Returns or sets a value indicating whether a Control-menu box is displayed on a form at run time. Read-only at run time.

Syntax

object.**ControlBox**

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

Settings

The **ControlBox** property settings are:

Setting	Description
True	(Default) Displays the Control-menu box.
False	Removes the Control-menu box.

Remarks

To display a Control-menu box, you must also set the form's **BorderStyle** property to 1 (Fixed Single), 2 (Sizable), or 3 (Fixed Dialog).

Both modal and modeless windows can include a Control-menu box.

The commands available at run time depend on the settings for related properties — for example, setting **MaxButton** and **MinButton** to **False** disables the Maximize and Minimize commands on the Control menu, but the Move and Close commands remain available.

Note Settings you specify for the **ControlBox**, **BorderStyle**, **MaxButton**, and **MinButton** properties aren't reflected in the form's appearance until run time.

DrawWidth Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDrawWidthC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDrawWidthX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDrawWidthA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDrawWidthS"}

Returns or sets the line width for output from graphics methods.

Syntax

object.**DrawWidth** [= *size*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
size	A numeric expression from 1 through 32,767. This value represents the width of the line in pixels. The default is 1; that is, 1 pixel wide.

Remarks

Increase the value of this property to increase the width of the line. If the **DrawWidth** property setting is greater than 1, **DrawStyle** property settings 1 through 4 produce a solid line (the **DrawStyle** property value isn't changed). Setting **DrawWidth** to 1 allows **DrawStyle** to produce the results shown in the **DrawStyle** property table.

FontBold, FontItalic, FontStrikethru, FontUnderline Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFontBoldC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontBoldX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontBoldA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontBoldS"}

Return or set font styles in the following formats: **Bold**, *Italic*, ~~Strikethru~~, and Underline.

Note The **FontBold, FontItalic, FontStrikethru,** and **FontUnderline** properties are included for use with the **CommonDialog** control and for compatibility with earlier versions of Visual Basic. For additional functionality, use the new **Font** object properties (not available for the **CommonDialog** control).

Syntax

object.**FontBold** [= *boolean*]
object.**FontItalic** [= *boolean*]
object.**FontStrikethru** [= *boolean*]
object.**FontUnderline** [= *boolean*]

The **FontBold**, **FontItalic**, **FontStrikethru**, and **FontUnderline** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying the font style as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default for **FontBold**, except with the **CommonDialog** control) Turns on the formatting in that style.
False	(Default for **FontItalic**, **FontStrikethru**, and **FontUnderline**, and **FontBold** with the **CommonDialog** control) Turns off the formatting in that style.

Remarks

Use these font properties to format text, either at design time using the Properties window or at run time using code. For **PictureBox** controls and **Form** and **Printer** objects, setting these properties doesn't affect graphics or text already drawn on the control or object. For all other controls, font changes take effect on screen immediately.

To use these properties with the **CommonDialog** control, the **Effects** flag must be set.

Note Fonts available in Visual Basic vary depending on your system configuration, display devices, and printing devices. Font-related properties can be set only to values for which actual fonts exist.

In general, you should change the **FontName** property before you set size and style attributes with the **FontSize**, **FontBold**, **FontItalic**, **FontStrikethru**, and **FontUnderline** properties. However, when you set TrueType fonts to smaller than 8 points, you should set the point size with the **FontSize** property, then set the **FontName** property, and then set the size again with the **FontSize** property. The Microsoft Windows operating environment uses a different font for TrueType fonts that are smaller than 8 points.

FontName Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFontNameC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontNameX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontNameA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontNameS"}

Returns or sets the font used to display text in a control or in a run-time drawing or printing operation.

Note The **FontName** property is included for use with the **CommonDialog** control and for compatibility with earlier versions of Visual Basic. For additional functionality, use the new **Font** object properties (not available for the **CommonDialog** control).

Syntax

object.**FontName** [= *font*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
font	A string expression specifying the font name to use.

Remarks

The default for this property is determined by the system. Fonts available with Visual Basic vary depending on your system configuration, display devices, and printing devices. Font-related properties can be set only to values for which fonts exist.

In general, you should change **FontName** before setting size and style attributes with the **FontSize**, **FontBold**, **FontItalic**, **FontStrikethru**, and **FontUnderline** properties.

Note At run time, you can get information on fonts available to the system through the **FontCount** and **Fonts** properties.

FontSize Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFontSizeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontSizeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontSizeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontSizeS"}

Returns or sets the size of the font to be used for text displayed in a control or in a run-time drawing or printing operation.

Note The **FontSize** property is included for use with the **CommonDialog** control and for compatibility with earlier versions of Visual Basic. For additional functionality, use the new **Font** object properties (not available for the **CommonDialog** control).

Syntax

object.**FontSize** [= *points*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
points	A numeric expression specifying the font size to use, in points.

Remarks

Use this property to format text in the font size you want. The default is determined by the system. To change the default, specify the size of the font in points.

The maximum value for **FontSize** is 2160 points.

Note Fonts available with Visual Basic vary depending on your system configuration, display devices, and printing devices. Font-related properties can be set only to values for which fonts exist.

In general, you should change the **FontName** property before you set size and style attributes with the **FontSize**, **FontBold**, **FontItalic**, **FontStrikethru**, and **FontUnderline** properties. However, when you set TrueType fonts to smaller than 8 points, you should set the point size with the **FontSize** property, then set the **FontName** property, and then set the size again with the **FontSize** property. The Microsoft Windows operating environment uses a different font for TrueType fonts that are smaller than 8 points.

Height, Width Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHeightC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHeightX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHeightA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHeightS"}

Return or set the dimensions of an object or the width of the **Columns** object of a **DBGrid** control. For the **Printer** and **Screen** objects, not available at design time.

Syntax

object.**Height** [= *number*]
object.**Width** [= *number*]

The **Height** and **Width** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression specifying the dimensions of an object, as described in Settings.

Settings

Measurements are calculated as follows:

- **Form** — the external height and width of the form, including the borders and title bar.
- **Control** — measured from the center of the control's border so that controls with different border widths align correctly. These properties use the scale units of a control's container.
- **Printer** object — the physical dimensions of the paper set up for the printing device; not available at design time. If set at run time, values in these properties are used instead of the setting of the **PaperSize** property.
- **Screen** object — the height and width of the screen; not available at design time and read-only at run time.
- **Picture** object — the height and width of the picture in HiMetric units.

Remarks

For **Form**, **Printer**, and **Screen** objects, these properties are always measured in twips. For a form or control, the values for these properties change as the object is sized by the user or by your code. Maximum limits of these properties for all objects are system-dependent.

If you set the **Height** and **Width** properties for a printer driver that doesn't allow these properties to be set, no error occurs and the size of the paper remains as it was. If you set **Height** and **Width** for a printer driver that allows only certain values to be specified, no error occurs and the property is set to whatever the driver allows. For example, you could set **Height** to 150 and the driver would set it to 144.

Use the **Height**, **Width**, **Left**, and **Top** properties for operations or calculations based on an object's total area, such as sizing or moving the object. Use the **ScaleLeft**, **ScaleTop**, **ScaleHeight**, and **ScaleWidth** properties for operations or calculations based on an object's internal area, such as drawing or moving objects within another object.

Note The **Height** property can't be changed for the **DriveListBox** control or for the **ComboBox** control, whose **Style** property setting is 0 (Dropdown Combo) or 2 (Dropdown List).

For the **Columns** object of the **DBGrid** control, **Width** is specified in the unit of measure of the object that contains the **DBGrid**. The default value for **Width** is the value of the **DefColWidth** property of **DBGrid**.

For the **Picture** object, use the **ScaleX** and **ScaleY** methods to convert HiMetric units into the scale you need.

Icon Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproIconC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproIconX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproIconA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproIconS"}

Returns the icon displayed when a form is minimized at run time.

Syntax

object.**Icon**

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

Remarks

Use this property to specify an icon for any form that the user can minimize at run time.

For example, you can assign a unique icon to a form to indicate the form's function. Specify the icon by loading it using the Properties window at design time. The file you load must have the .ico filename extension and format. If you don't specify an icon, the Visual Basic default icon for forms is used.

You can use the Visual Basic Icon Library (in the Icons subdirectory) as a source for icons. When you create an executable file, you can assign an icon to the application by using the **Icon** property of any form in that application.

Note You can see a form's icon in Windows 95 in the upper left corner of the form, or when the form is minimized in both Windows 95 and Windows NT. If the form is minimized, the **BorderStyle** property must be set to either 1 (Fixed Single) or 2 (Sizable) and the **MinButton** property must be set to **True** for the icon to be visible.

At run time, you can assign an object's **Icon** property to another object's **DragIcon** or **Icon** property. You can also assign an icon returned by the **LoadPicture** function. Using **LoadPicture** without an argument assigns an empty (null) icon to the form, which enables you to draw on the icon at run time.

Interval Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproIntervalC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproIntervalX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproIntervalA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproIntervalS"}

Returns or sets the number of milliseconds between calls to a **Timer** control's Timer event.

Syntax

object.**Interval** [= *milliseconds*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
milliseconds	A numeric expression specifying the number of milliseconds, as described in Settings.

Settings

The settings for *milliseconds* are:

Setting	Description
0	(Default) Disables a **Timer** control.
1 to 65,535	Sets an interval (in milliseconds) that takes effect when a **Timer** control's **Enabled** property is set to **True**. For example, a value of 10,000 milliseconds equals 10 seconds. The maximum, 65,535 milliseconds, is equivalent to just over 1 minute.

Remarks

You can set a **Timer** control's **Interval** property at design time or run time. When using the **Interval** property, remember:

- The **Timer** control's **Enabled** property determines whether the control responds to the passage of time. Set **Enabled** to **False** to turn a **Timer** control off, and to **True** to turn it on. When a **Timer** control is enabled, its countdown always starts from the value of its **Interval** property setting.
- Create a Timer event procedure to tell Visual Basic what to do each time the **Interval** has passed.

Left, Top Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLeftC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLeftX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLeftA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLeftS"}

- **Left** — returns or sets the distance between the internal left edge of an object and the left edge of its container.
- **Top** — returns or sets the distance between the internal top edge of an object and the top edge of its container.

Syntax

object.**Left** [= *value*]
object.**Top** [= *value*]

The **Left** and **Top** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A numeric expression specifying distance.

Remarks

For a form, the **Left** and **Top** properties are always expressed in twips; for a control, they are measured in units depending on the coordinate system of its container. The values for these properties change as the object is moved by the user or by code. For a **Timer** control, these properties aren't available at run time.

For both properties, you can specify a single-precision number.

Use the **Left**, **Top**, **Height**, and **Width** properties for operations based on an object's external dimensions, such as moving or resizing. Use the **ScaleLeft**, **ScaleTop**, **ScaleHeight**, and **ScaleWidth** properties for operations based on an object's internal dimensions, such as drawing or moving objects that are contained within the object. The scale-related properties apply only to **PictureBox** controls and **Form** and **Printer** objects.

List Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproListC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproListX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproListA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproListS"}

Returns or sets the items contained in a control's list portion. The list is a string array in which each element is a list item. Available at design time for **ListBox** and **ComboBox** controls through the property browser; read-only at run time for **DirListBox**, **DriveListBox**, and **FileListBox** controls; read/write at run time for **ComboBox** and **ListBox** controls.

Syntax

object.**List**(*index*) [= *string*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	The number of a specific item in the list.
string	A string expression specifying the list item.

Remarks

Use this property to access list items.

For all controls except the **DirListBox**, the index of the first item is 0 and the index of the last item is **ListCount**–1.

For a **DirListBox** control, the index number sequence is based on the current directories and subdirectories when the control is created at run time. The directory that is currently expanded is represented using the index–1. Directories above the currently expanded directory are represented by negative indexes with higher absolute values. For example,–2 is the parent directory of the directory that is currently expanded and–3 is the directory above that. Directories below the directory that is currently expanded range from 0 to **ListCount**–1.

Initially, **ComboBox** and **ListBox** controls contain an empty list. For the file-system controls, the list is based on conditions that exist when the control is created at run time:

- **DirListBox** — contains a list of all directories, using the range *-n* to **ListCount**–*1*.
- **DriveListBox** — contains the list of drive connections in effect.
- **FileListBox** — contains the list of files in the directory that is currently expanded that match the **Pattern** property. The path isn't included.

The **List** property works in conjunction with the **ListCount** and **ListIndex** properties.

For all applicable controls except a **DirListBox**, enumerating a list from 0 to **ListCount** -1 returns all items in the list. For a **DirListBox** control, enumerating the list from–*n* to **ListCount**–1 returns a list containing all directories and subdirectories visible from the directory that is currently expanded. In this case *n* is the number of directory levels above the directory that is currently expanded.

Note To specify items you want to display in a **ComboBox** or **ListBox** control, use the **AddItem** method. To remove items, use the **RemoveItem** method. To keep items in alphabetic order, set the control's **Sorted** property to **True** before adding items to the list.

Using an `Option Base = 1` statement in the Declarations section doesn't affect the enumeration of elements in Visual Basic controls. The first element is always 0.

When the List index is outside the range of actual entries in the list box, a zero-length string ("") is returned. For example, `List(-1)` returns a zero-length string for a **ComboBox** or **ListBox** control.

ListCount Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproListCountC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproListCountX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproListCountA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproListCountS"}

Returns the number of items in the list portion of a control.

Syntax

object.**ListCount**

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

Remarks

ListCount provides specific information for each control:

- **ComboBox** and **ListBox** controls — the number of items in the list.
- **DirListBox** control — the number of subdirectories in the current directory.
- **DriveListBox** control — the number of drive connections.
- **FileListBox** control — the number of files in the current directory that match the **Pattern** property setting.

If no item is selected, the **ListIndex** property value is –1. The first item in the list is **ListIndex** = 0, and **ListCount** is always one more than the largest **ListIndex** value.

ListIndex Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproListIndexC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproListIndexX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproListIndexA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproListIndexS"}

Returns or sets the index of the currently selected item in the control. Not available at design time.

Syntax

object.**ListIndex** [= *index*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	A numeric expression specifying the index of the current item, as described in Settings.

Settings

The settings for *index* are:

Setting	Description
–1	(Default for **ComboBox**, **DirListBox**, and **DriveListBox** controls) Indicates no item is currently selected. For a **ComboBox** control, indicates the user has entered new text into the text box portion. For a **DirListBox** control, indicates the index of the current path. For a **DriveListBox** control, indicates the index of the current drive when the control is created at run time.
n	(Default for **FileListBox** and **ListBox** controls) A number indicating the index of the currently selected item.

Remarks

The expression `List(List1.ListIndex)` returns the string for the currently selected item.

The first item in the list is **ListIndex** = 0, and **ListCount** is always one more than the largest **ListIndex** value.

For a control in which users can make multiple selections, this property's behavior depends on the number of items selected. If only one item is selected, **ListIndex** returns the index of that item. In a multiple selection, **ListIndex** returns the index of the item contained within the focus rectangle, whether or not that item is actually selected.

MaxButton Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMaxButtonC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMaxButtonX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMaxButtonA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMaxButtonS"}

Returns a value indicating whether a form has a Maximize button.

Syntax

object.**MaxButton**

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

Settings

The **MaxButton** property settings are:

Setting	Description
True	(Default) The form has a Maximize button.
False	The form doesn't have a Maximize button.

Remarks

A Maximize button enables users to enlarge a form window to full-screen size. To display a Maximize button, you must also set the form's **BorderStyle** property to either 1 (Fixed Single), 2 (Sizable), or 3 (Fixed Double).

A Maximize button automatically becomes a Restore button when a window is maximized. Minimizing or restoring a window automatically changes the Restore button back to a Maximize button.

The settings you specify for the **MaxButton**, **MinButton**, **BorderStyle**, and **ControlBox** properties aren't reflected in the form's appearance until run time.

Note Maximizing a form at run time generates a Resize event. The **WindowState** property reflects the current state of the window. If you set the **WindowState** property to 2 (Maximized), the form is maximized independently of whatever settings are in effect for the **MaxButton** and **BorderStyle** properties.

MinButton Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMinButtonC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMinButtonX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMinButtonA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMinButtonS"}

Returns a value indicating whether a form has a Minimize button.

Syntax

object.**MinButton**

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

Return Values

The **MinButton** return values are:

Setting	Description
True	(Default) The form has a Minimize button.
False	The form doesn't have a Minimize button.

Remarks

A Minimize button enables users to minimize a form window to an icon. To display a Minimize button, you must also set the form's **BorderStyle** property to either 1 (Fixed Single), 2 (Sizable), or 3 (Fixed Double).

The settings you specify for the **MaxButton**, **MinButton**, **BorderStyle**, and **ControlBox** properties aren't reflected in the form's appearance until run time.

Note Minimizing a form to an icon at run time generates a Resize event. The **WindowState** property reflects the current state of the window. If you set the **WindowState** property to 2 (Maximized), the form is maximized independently of whatever settings are in effect for the **MaxButton** and **BorderStyle** properties.

Picture Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPictureC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPictureX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPictureA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPictureS"}

Returns or sets a graphic to be displayed in a control. For the **OLE** container control, not available at design time and read-only at run time.

Syntax

object.**Picture** [= *picture*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
picture	A string expression specifying a file containing a graphic, as described in Settings.

Settings

The settings for *picture* are:

Setting	Description
(None)	(Default) No picture.
(Bitmap, icon, metafile)	Specifies a graphic. You can load the graphic from the Properties window at design time. At run time, you can also set this property using the **LoadPicture** function on a bitmap, icon, or metafile.

Remarks

At design time, you can transfer a graphic with the Clipboard using the Copy, Cut, and Paste commands on the Edit menu. At run time, you can use Clipboard methods such as **GetData**, **SetData**, and **GetFormat** with the nontext Clipboard constants **vbCFBitmap**, **vbCFMetafile**, and **vbCFDIB**, which are listed in the Visual Basic (VB) object library in the Object Browser.

When setting the **Picture** property at design time, the graphic is saved and loaded with the form. If you create an executable file, the file contains the image. When you load a graphic at run time, the graphic isn't saved with the application. Use the **SavePicture** statement to save a graphic from a form or picture box into a file.

Note At run time, the **Picture** property can be set to any other object's **DragIcon**, **Icon**, **Image**, or **Picture** property, or you can assign it the graphic returned by the **LoadPicture** function.

Sorted Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSortedC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSortedX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSortedA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSortedS"}

Returns a value indicating whether the elements of a control are automatically sorted alphabetically.

Syntax

object.**Sorted**

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

Return Values

The **Sorted** property return values are:

Setting	Description
True	List items are sorted alphabetically (case-insensitive).
False	(Default) List items aren't sorted alphabetically.

Remarks

When this property is **True**, Visual Basic handles almost all necessary string processing to maintain alphabetic order, including changing the index numbers for items as required by the addition or removal of items.

Note Using the **AddItem** method to add an element to a specific location in the list may violate the sort order, and subsequent additions may not be correctly sorted.

TabIndex Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTabIndEXC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTabIndEXX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTabIndexA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTabIndexS"}

Returns or sets the tab order of most objects within their parent form.

Syntax

object.**TabIndex** [= *index*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer from 0 to (*n*–1), where *n* is the number of controls on the form that have a **TabIndex** property. Assigning a **TabIndex** value of less than 0 generates an error.

Remarks

By default, Visual Basic assigns a tab order to controls as you draw them on a form, with the exception of the **Menu**, **Timer**, **Data**, **Image**, **Line** and **Shape** controls, which are not included in the tab order. At run time, invisible or disabled controls and controls that can't receive the focus (**Frame** and **Label** controls) remain in the tab order but are skipped during tabbing.

Each new control is placed last in the tab order. If you change the value of a control's **TabIndex** property to adjust the default tab order, Visual Basic automatically renumbers the **TabIndex** of other controls to reflect insertions and deletions. You can make changes at design time using the Properties window or at run time in code.

The **TabIndex** property isn't affected by the **ZOrder** method.

Note A control's tab order doesn't affect its associated access key. If you press the access key for a **Frame** or **Label** control, the focus moves to the next control in the tab order that can receive the focus.

When loading forms saved as ASCII text, controls with a **TabIndex** property that aren't listed in the form description are automatically assigned a **TabIndex** value. In subsequently loaded controls, if existing **TabIndex** values conflict with earlier assigned values, the controls are automatically assigned new values.

When you delete one or more controls, you can use the **Undo** command to restore the controls and all their properties except for the **TabIndex** property, which can't be restored. **TabIndex** is reset to the end of the tab order when you use Undo.

Tag Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTagC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTagX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTagA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTagS"}

Returns or sets an expression that stores any extra data needed for your program. Unlike other properties, the value of the **Tag** property isn't used by Visual Basic; you can use this property to identify objects.

Syntax

object.**Tag** [= *expression*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
expression	A string expression identifying the object. The default is a zero-length string ("").

Remarks

You can use this property to assign an identification string to an object without affecting any of its other property settings or causing side effects. The **Tag** property is useful when you need to check the identity of a control or **MDIForm** object that is passed as a variable to a procedure.

Tip When you create a new instance of a form, assign a unique value to the **Tag** property.

Text Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTextC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTextX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTextA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTextS"}

- **ComboBox** control (**Style** property set to 0 [Dropdown Combo] or to 1 [Simple Combo]) and **TextBox** control — returns or sets the text contained in the edit area.
- **ComboBox** control (**Style** property set to 2 [Dropdown List]) and **ListBox** control — returns the selected item in the list box; the value returned is always equivalent to the value returned by the expression `List(ListIndex)`. Read-only at design time; read-only at run time.

Syntax

object.**Text** [= *string*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	A string expression specifying text.

Remarks

At design time only, the defaults for the **Text** property are:

- **ComboBox** and **TextBox** controls — the control's **Name** property.
- **ListBox** control — a zero-length string ("").

For a **ComboBox** with the **Style** property set to 0 (Dropdown Combo) or to 1 (Simple Combo) or for a **TextBox**, this property is useful for reading the actual string contained in the edit area of the control. For a **ComboBox** or **ListBox** control with the **Style** property set to 2 (Dropdown List), you can use the **Text** property to determine the currently selected item.

The **Text** setting for a **TextBox** control is limited to 2048 characters unless the **MultiLine** property is **True**, in which case the limit is about 32K.

Value Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproValueC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproValueX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproValueA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproValueS"}

- **CheckBox** and **OptionButton** controls — returns or sets the state of the control.
- **CommandButton** control — returns or sets a value indicating whether the button is chosen; not available at design time.
- **Field** object — returns or sets the content of a field; not available at design time.
- **HScrollBar** and **VScrollBar** controls (horizontal and vertical scroll bars) — returns or sets the current position of the scroll bar, whose return value is always between the values for the **Max** and **Min** properties, inclusive.

Syntax

object.**Value** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	Value specifying the state, content, or position of a control, as described in Settings.

Settings

The settings for *value* are:

- **CheckBox** control — 0 is Unchecked (default), 1 is Checked, and 2 is Grayed (dimmed).
- **CommandButton** control — **True** indicates the button is chosen; **False** (default) indicates the button isn't chosen. Setting the **Value** property to **True** in code invokes the button's Click event.
- **Field** object — restricted only by the Field data types.
- **HScrollBar** and **VScrollBar** controls — set values between –32,768 and 32,767 to position the scroll box.
- **OptionButton** control — **True** indicates the button is selected; **False** (default) indicates the button isn't selected.

Remarks

A default property of an object is assumed, and doesn't need to be specified in code. For example, **Field** is the default property of any **Recordset**, and **Value** is the default property of a **Field** object. This makes the two statements below equivalent:

```
Dn.Fields("PubID").Value = X
Dn.("PubID") = X
```

The first statement *specifies* the default properties; the second statement *assumes* them.

Visible Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproVisibleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproVisibleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproVisibleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproVisibleS"}

Returns or sets a value indicating whether an object is visible or hidden.

Syntax

object.**Visible** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying whether the object is visible or hidden.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) Object is visible.
False	Object is hidden.

Remarks

To hide an object at startup, set the **Visible** property to **False** at design time. Setting this property in code enables you to hide and later redisplay a control at run time in response to a particular event.

Note Using the **Show** or **Hide** method on a form is the same as setting the form's **Visible** property in code to **True** or **False**, respectively.

BackColor, ForeColor Properties Example

This example resets foreground and background colors randomly twice each second for a form and **PictureBox** control. To try this example, paste the code into the Declarations section of a form that contains a **PictureBox** control and a **Timer** control, and then press F5.

```
Private Sub Form_Load ()
   Timer1.Interval = 500
End Sub

Private Sub Timer1_Timer ()
   BackColor = QBColor(Rnd * 15)
   ForeColor = QBColor(Rnd * 10)
   Picture1.BackColor = QBColor(Rnd * 15)
   Picture1.ForeColor = QBColor(Rnd * 10)
End Sub
```

BorderWidth Property Example

This example uses two **ComboBox** controls to select different widths and styles for the borders of a **Shape** control. To try this example, paste the code into the Declarations section of a form that contains a **Shape** control and one **ComboBox** control. For the **ComboBox**, set **Style** = 2 and **Index** = 0 (to create a control array), and then press F5 and click the form.

```
Private Sub Form_Load ()
   Combo1(0).Width = 1440 * 1.5
   Load Combo1(1)
   Combo1(1).Top = Combo1(0).Top + Combo1(0).Height * 1.5
   Combo1(1).Visible = True
   For I = 0 To 6
      Combo1(0).AddItem "BorderStyle = " & I
   Next I
   For I = 1 To 10
      Combo1(1).AddItem "BorderWidth = " & I
   Next I
   Combo1(0).ListIndex = 1
   Combo1(1).ListIndex = 0
End Sub

Private Sub Combo1_Click (Index As Integer)
   If Index = 0 Then
      Shape1.BorderStyle = Combo1(0).ListIndex
   Else
      Shape1.BorderWidth = Combo1(1).ListIndex + 1
   End If
End Sub
```

Col, Row Properties Example

This example puts "Here" into the current cell and then changes the active cell to the third cell in the third row and puts "There" into that cell. To try this example, use the Components dialog box to add an **MS Flex Grid** control to the toolbox (from the Project menu, choose Components, and then check Microsoft Flex Grid Control), and then draw a grid on a new form. To run the program, press F5, and then click the grid.

```
Private Sub Form_Load ()
   MSFlexGrid1.Rows = 8 ' Set rows and columns.
   MSFlexGrid1.Cols = 5
End Sub

Private Sub MSFlexGrid1_Click ()
   ' Put text in current cell.
   MSFlexGrid1.Text = "Here"
   ' Put text in third row, third column.
   MSFlexGrid1.Col = 2
   MSFlexGrid1.Row = 2
   MSFlexGrid1.Text = "There"
End Sub
```

The next example displays the location of the active cell and the range of the selection as a user selects a cell or range of cells. Notice that when selecting a range, the active cell doesn't change. Select a range, and then click the form to move the active cell around the perimeter of the selection. Notice that the selected range doesn't change.

To try this example, create a new project, add an **MS Flex Grid** control using the Components dialog box (from the Project menu, choose Components, and then check Microsoft Flex Grid Control), and then draw an **MS Flex Grid** and two labels. Copy the code into the Declarations section, and then press F5 to run the program.

```
Private Sub Form_Load ()
   MSFlexGrid1.Cols = 6 ' Set columns and rows.
   MSFlexGrid1.Rows = 7
End Sub

Private Sub MSFlexGrid1_RowColChange ()
   Msg = "Active Cell: " & Chr(64 + MSFlexGrid1.Col)
   Mst = Msg & MSFlexGrid1.Row
   Label1.Caption = Msg
End Sub

Private Sub MSFlexGrid1_SelChange ()
   Msg = "Selection: " & Chr(64 + MSFlexGrid1.SelStartCol)
   Msg = Msg & MSFlexGrid1.SelStartRow
   Msg = Msg & ":" & Chr(64 + MSFlexGrid1.SelEndCol)
   Msg = Msg & MSFlexGrid1.SelEndRow
   Label2.Caption = Msg
End Sub

Private Sub Form_Click ()
   ' This procedure moves the active cell around
   ' the perimeter of the selected range
   ' of cells with each click on the form.
   Dim GR, GC As Integer
   If MSFlexGrid1.Row = MSFlexGrid1.SelStartRow Then
      If MSFlexGrid1.Col = MSFlexGrid1.SelEndCol Then
         GR = 1: GC = 0
      Else
```

```
            GR = 0: GC = 1
         End If
      ElseIf MSFlexGrid1.Row = MSFlexGrid1.SelEndRow Then
         If MSFlexGrid1.Col = MSFlexGrid1.SelStartCol Then
            GR = -1: GC = 0
         Else
            GR = 0: GC = -1
         End If
      Else
         If MSFlexGrid1.Col = MSFlexGrid1.SelStartCol Then
            GR = -1: GC = 0
         Else
            GR = 1: GC = 0
         End If
      End If
      MSFlexGrid1.Row = MSFlexGrid1.Row + GR
      MSFlexGrid1.Col = MSFlexGrid1.Col + GC
End Sub
```

DrawWidth Property Example

This example draws a gradually thickening line across a form. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim I ' Declare variable.
   DrawWidth = 1  ' Set starting pen width.
   PSet (0, ScaleHeight / 2) ' Set starting point.
   ForeColor = QBColor(5)  ' Set pen color.
   For I = 1 To 100 Step 10  ' Set up loop.
      DrawWidth = I  ' Reset pen width.
      Line - Step(ScaleWidth / 10, 0)    ' Draw a line.
   Next I
End Sub
```

FontBold, FontItalic, FontStrikethru, FontUnderline Properties Example

This example puts text on a form in one of two combinations of styles with each mouse click. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   FontStrikethru = Not FontStrikethru   ' Toggle strikethrough.
   FontItalic = Not FontItalic     ' Toggle font style.
   Print "Now is the time!"  ' Print some text.
End Sub
```

FontName Property Example

This example prints the name of each font using the particular font. To try this example, paste the code into the Declarations section of a form. Press F5 to run the program, and then click the form. Each time you click the form, the font name is printed.

```
Private Sub Form_Click ()
   Static I ' Declare variables.
   Dim OldFont
   OldFont = FontName    ' Preserve original font.
   FontName = Screen.Fonts(I)       ' Change to new font.
   Print Screen.Fonts(I)    ' Print name of font.
   I = I + 1    ' Increment counter.
   If I = FontCount Then I = 0      ' Start over.
   FontName = OldFont    ' Restore original font.
End Sub
```

FontSize Property Example

This example prints text on your form in two different point sizes with each click of the mouse. To try this example, paste the code into the Declarations section of a form. Press F5 to run the program, and then click the form.

```
Private Sub Form_Click ()
   FontSize = 24  ' Set FontSize.
   Print "This is 24-point type."  ' Print large type.
   FontSize = 8   ' Set FontSize.
   Print "This is 8-point type."   ' Print small type.
End Sub
```

Height, Width Properties Example

This example sets the size of a form to 75 percent of screen size and centers the form when it is loaded. To try this example, paste the code into the Declarations section of a form. Then press F5 and click the form.

```
Private Sub Form_Click ()
   Width = Screen.Width * .75      ' Set width of form.
   Height = Screen.Height * .75    ' Set height of form.
   Left = (Screen.Width - Width) / 2     ' Center form horizontally.
   Top = (Screen.Height - Height) / 2    ' Center form vertically.
End Sub
```

Icon Property Example

This example creates a blank icon for a form and draws colored dots on the icon as long as the form is minimized. To try this example, paste the code into the Declarations section of a form, and then press F5 and minimize the form.

Note This example works only with Windows NT 3.5x.

```
Private Sub Form_Resize ()
   Dim X, Y ' Declare variables.
   If Form1.WindowState = vbMinimized Then
      Form1.Icon = LoadPicture()    ' Load a blank icon.
      Do While Form1.WindowState = vbMinimized ' While form is minimized,
         Form1.DrawWidth = 10       ' set size of dot.
         ' Choose random color for dot.
         Form1.ForeColor = QBColor(Int(Rnd * 15))
         ' Set random location on icon.
         X = Form1.Width * Rnd
         Y = Form1.Height * Rnd
         PSet (X, Y) ' Draw dot on icon.
         DoEvents ' Allow other events.
      Loop
   End If
End Sub
```

Interval Property Example

This example enables you to adjust the speed at which a form switches colors. To try this example, paste the code into the Declarations section of a form that contains a **Timer** control, an **HScrollBar** control (horizontal scroll bar), and a **PictureBox** control, and then press F5 and click the scroll bar.

```
Private Sub Form_Load ()
   Timer1.Interval = 900   ' Set interval.
   HScroll1.Min = 100   ' Set minimum.
   HScroll1.Max = 900   ' Set maximum.
End Sub
Private Sub HScroll1_Change ()
   ' Set interval according to scroll bar value.
   Timer1.Interval = 1000 - HScroll1.Value
End Sub
Private Sub Timer1_Timer ()
   ' Switch BackColor between red and blue.
   If Picture1.BackColor = RGB(255, 0, 0) Then
      Picture1.BackColor = RGB(0, 0, 255)
   Else
      Picture1.BackColor = RGB(255, 0, 0)
   End If
End Sub
```

Left, Top Properties Example

This example sets the size of a form to 75 percent of screen size and centers the form when it's loaded. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Width = Screen.Width * .75      ' Set width of form.
   Height = Screen.Height * .75    ' Set height of form.
   Left = (Screen.Width - Width) / 2     ' Center form horizontally.
   Top = (Screen.Height - Height) / 2    ' Center form vertically.
End Sub
```

List Property Example

This example loads a **ComboBox** control with a list of sandwich names and displays the first item in the list. To try this example, paste the code into the Declarations section of a form that contains a **ComboBox** control, and then press F5.

```
Private Sub Form_Load ()
   Combo1.AddItem "Denver Sandwich"      ' Add each item to list.
   Combo1.AddItem "Reuben Sandwich"
   Combo1.AddItem "Turkey Sandwich"
   Combo1.Text = Combo1.List(0)    ' Display first item.
End Sub
```

ListCount Property Example

This example loads a list of your printer fonts into a **ComboBox** control, displays the first item in the list, and prints the total number of fonts. Each click of the command button changes all items in the list to uppercase or lowercase. To try this example, paste the code into the Declarations section of a form that contains a **ComboBox** control (**Style** = 2) and a **CommandButton** control, and then press F5 and click the **CommandButton**.

```
Private Sub Form_Load ()
   Dim I ' Declare variable.
   AutoRedraw = True ' Set AutoRedraw.
   For I = 0 To Printer.FontCount - 1    ' Put font names in list.
      Combo1.AddItem Printer.Fonts(I)
   Next I
   Combo1.ListIndex = 0 ' Set text to first item.
   ' Print ListCount information on form.
   Print "Number of printer fonts: "; Combo1.ListCount
End Sub
Private Sub Command1_Click ()
   Static UpperCase
   Dim I ' Declare variable.
   For I = 0 To Combo1.ListCount - 1     ' Loop through list.
      If UpperCase Then
         Combo1.List(I) = UCase(Combo1.List(I))
      Else
         Combo1.List(I) = LCase(Combo1.List(I))
      End If
   Next I
   UpperCase = Not UpperCase ' Change case.
End Sub
```

ListIndex Property Example

This example displays the names of three players in a **ListBox** control and the corresponding salary of the selected player in a **Label** control. To try this example, paste the code into the Declarations section of a form that contains a **ComboBox** control and a **Label** control, and then press F5 and choose a name from the **ComboBox**.

```
Dim Player(0 To 2)' Dimension two arrays.
Dim Salary(0 To 2)
Private Sub Form_Load ()
   Dim I ' Declare variable.
   AutoSize = True
   Player(0) = "Miggey McMoo"      ' Enter data into arrays.
   Player(1) = "Alf Hinshaw"
   Player(2) = "Woofer Dean"
   Salary(0) = "$234,500"
   Salary(1) = "$158,900"
   Salary(2) = "$1,030,500"
   For I = 0 To 2 ' Add names to list.
      Combo1.AddItem Player(I)
   Next I
   Combo1.ListIndex = 0 ' Display first item in list.
End Sub

Private Sub Combo1_Click ()
   ' Display corresponding salary for name.
   Label1.Caption = Salary(Combo1.ListIndex)
End Sub
```

Picture Property Example

This example loads icons from the Visual Basic icon library into two of three **PictureBox** controls. When you click the form, the third **PictureBox** is used to switch the icons. You can use any two icons. Paste the code into the Declarations section of a form that has three small **PictureBox** controls (for `Picture3`, set **Visible = False**). Press F5 to run the program, and then click the form.

```
Private Sub Form_Load ()
   ' Load the icons.
   Picture1.Picture = LoadPicture("ICONS\COMPUTER\TRASH02A.ICO")
   Picture2.Picture = LoadPicture("ICONS\COMPUTER\TRASH02B.ICO")
End Sub

Private Sub Form_Click ()
   ' Switch the icons.
   Picture3.Picture = Picture1.Picture
   Picture1.Picture = Picture2.Picture
   Picture2.Picture = Picture3.Picture
   ' Clear the third picture (not necessary if not visible).
   Picture3.Picture = LoadPicture()
End Sub
```

This example pastes a bitmap from the Clipboard into a **PictureBox** control. To find the value of Clipboard format constants (starting with **vbCF**), see the Visual Basic (VB) object library in the Object Browser. To try this example, paste the code into the Declarations section of a form that has a **PictureBox** control. Press F5, and then in another application, copy an icon onto the Clipboard, switch to Visual Basic, and click the form.

```
Private Sub Form_Click ()
   Picture1.Picture = Clipboard.GetData(vbCFDIB)
End Sub
```

TabIndex Property Example

This example reverses the tab order of a group of buttons by changing the **TabIndex** property of a command button array. To try this example, paste the code into the Declarations section of a form that contains four **CommandButton** controls. Set the **Name** property to CommandX for each button to create the control array, and then press F5 and click the form to reverse the tab order of the buttons.

```
Private Sub Form_Click ()
   Dim I, X ' Declare variables.
   ' Reverse tab order by setting start value of X.
   If CommandX(0).TabIndex = 0 Then X = 4 Else X = 1
      For I = 0 To 3
         CommandX(I).Caption = X   ' Set caption.
         CommandX(I).TabIndex = X - 1    ' Set tab order.
         If CommandX(0).TabIndex = 3 Then
            X = X - 1   ' Decrement X.
         Else
            X = X + 1   ' Increment X.
         End If
      Next I
End Sub
```

Tag Property Example

This example displays a unique icon for each control being dragged. To try this example, paste the code into the Declarations section of a form that contains three **PictureBox** controls. Set the **DragMode** property to 1 for Picture1 and Picture2, and then press F5. Use the mouse to drag Picture1 or Picture2 over Picture3 controls.

```
Private Sub Form_Load ()
    Picture1.Tag = "ICONS\ARROWS\POINT03.ICO"
    Picture2.Tag = "ICONS\ARROWS\POINT04.ICO"
End Sub

Private Sub Picture3_DragOver (Source As Control, X As Single, Y As Single,
State As Integer)
   If State = vbEnter Then
      ' Select based on each PictureBox's Name property.
      Select Case Source.Name
      Case "Picture1"
         ' Load icon for Picture1.
         Source.DragIcon = LoadPicture(Picture1.Tag)             Case
"Picture2"
         ' Load icon for Picture2.
         Source.DragIcon = LoadPicture(Picture2.Tag)
      End Select
   ElseIf State = vbLeave Then
      ' When source isn't over Picture3, unload icon.
      Source.DragIcon = LoadPicture ()
   End If
End Sub
```

Text Property Example

This example illustrates the **Text** property. To try this example, paste the code into the Declarations section of a form that contains three **TextBox** controls and a **CommandButton** control, and then press F5 and enter text in Text1.

```
Private Sub Text1_Change ()
   Text2.Text = LCase(Text1.Text)  ' Display text as lowercase.
   Text3.Text = UCase(Text1.Text)  ' Display text as uppercase.
End Sub

Private Sub Command1_Click ()' Delete text.
   Text1.Text = ""
End Sub
```

Value Property Example

This example displays an **HScrollBar** (horizontal scroll bar) control's numeric value in a **TextBox** control. To try this example, paste the code into the Declarations section of a form that has a **TextBox** control and an **HScrollBar** control. Press F5 to run the program, and then click the scroll bar.

```
Private Sub Form_Load ()
   HScroll1.Min = 0  ' Initialize scroll bar.
   HScroll1.Max = 1000
   HScroll1.LargeChange = 100
   HScroll1.SmallChange = 1
End Sub

Private Sub HScroll1_Change ()
   Text1.Text = Format (HScroll1.Value)
End Sub
```

Visible Property Example

This example creates animation using two **PictureBox** controls. To try this example, paste the code into the Declarations section of a form that contains two icon-sized **PictureBox** controls. Set the **Name** property to FileCab for both **PictureBox** controls to create an array, and then press F5 and click the picture to view the animation.

```
Private Sub Form_Load ()
   Dim I ' Declare variable.
   FileCab(0).BorderStyle = 0      ' Set BorderStyle.
   FileCab(1).BorderStyle = 0
   ' Load icons into picture boxes.
   FileCab(1).Picture = LoadPicture("ICONS\OFFICE\FILES03B.ICO")
   FileCab(0).Picture = LoadPicture("ICONS\OFFICE\FILES03A.ICO")
   For I = 0 To 1
      FileCab(I).Move 400, 400     ' Place graphics at same spot.
   Next I
   FileCab(1).Visible = False      ' Set to invisible.
   FileCab(0).Visible = True ' Set to visible.
End Sub

Private Sub FileCab_Click (Index As Integer)
   Dim I ' Declare variable.
   For I = 0 To 1
      ' Switch the visibility for both graphics.
      FileCab(I).Visible = Not FileCab(I).Visible
   Next I
End Sub
```

Align Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAlignC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAlignX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAlignA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAlignS"}

Returns or sets a value that determines whether an object is displayed in any size anywhere on a form or whether it's displayed at the top, bottom, left, or right of the form and is automatically sized to fit the form's width.

Syntax

object.**Align** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	An integer that specifies how an object is displayed, as described in Settings.

Settings

The settings for *number* are:

Constant	Setting	Description
vbAlignNone	0	(Default in a non-MDI form) None — size and location can be set at design time or in code. This setting is ignored if the object is on an MDI form.
VbAlignTop	1	(Default in an MDI form) Top — object is at the top of the form, and its width is equal to the form's **ScaleWidth** property setting.
VbAlignBottom	2	Bottom — object is at the bottom of the form, and its width is equal to the form's **ScaleWidth** property setting.
VbAlignLeft	3	Left — object is at the left of the form, and its width is equal to the form's **ScaleWidth** property setting.
VbAlignRight	4	Right — object is at the right of the form, and its width is equal to the form's **ScaleWidth** property setting.

Remarks

You can use the **Align** property to quickly create a toolbar or status bar at the top or bottom of a form. As a user changes the size of the form, an object with **Align** set to 1 or 2 automatically resizes to fit the width of the form.

PictureBox and **Data** controls are the only standard controls that can be placed on an MDI form. The internal area of an MDI form is defined by the space not covered by controls. When an MDI child form is maximized within the parent MDI form, it won't cover any controls.

Use *number* settings 3 and 4 to align toolbars at the left and right sides of a form or MDI form. If there are two toolbars in a corner of an MDI form, the top- or bottom-aligned one extends to the corner, taking precedence over the left- or right-aligned one. Left- and right-aligned objects occupy the

internal area on an MDI form, just like top- and bottom-aligned objects.

Alignment Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAlignmentC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAlignmentX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAlignmentA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAlignmentS"}

Returns or sets a value that determines the alignment of a **CheckBox** or **OptionButton** control, text in a control, or values in a column of a **DBGrid** control. Read-only at run time for **CheckBox**, **OptionButton**, and **TextBox** controls.

Syntax

object.**Alignment** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
Number	An integer that specifies the type of alignment, as described in Settings.

Settings

For **CheckBox** and **OptionButton** controls, the settings for *number* are:

Constant	Setting	Description
vbLeftJustify	0	(Default) Text is left-aligned; control is right-aligned.
VbRightJustify	1	Text is right-aligned; control is left-aligned.

For **Label** and **TextBox** controls, the settings for *number* are:

Constant	Setting	Description
vbLeftJustify	0	(Default) Text is left-aligned.
VbRightJustify	1	Text is right-aligned.
VbCenter	2	Text is centered.

For a **DBGrid** column, the settings for *number* are:

Constant	Setting	Description
dbgLeft	0	Text is left-aligned.
DbgRight	1	Text is right-aligned.
DbgCenter	2	Text is centered.
DbgGeneral	3	(Default) General — Text is left-aligned; numbers are right-aligned.

Remarks

You can display text to the right or left of **OptionButton** and **CheckBox** controls. By default, text is left-aligned.

The **MultiLine** property in a **Textbox** control must be set to **True** for the **Alignment** property to work correctly. If the **MultiLine** property setting of a **TextBox** control is **False**, the **Alignment** property is ignored.

Archive, Hidden, Normal, System Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproArchiveC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproArchiveX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproArchiveA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproArchiveS"}

Return or set a value that determines whether a **FileListBox** control displays files with Archive, Hidden, Normal, or System attributes.

Syntax

object.**Archive** [= *boolean*]
object.**Hidden** [= *boolean*]
object.**Normal** [= *boolean*]
object.**System** [= *boolean*]

The **Archive**, **Hidden**, **Normal**, and **System** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that specifies the type of files displayed, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default for Archive and Normal) Displays files with the property's attribute in the **FileListBox** control.
False	(Default for Hidden and System) Displays files without the property's attribute in the **FileListBox** control.

Remarks

Use these properties to specify the type of files to display in a **FileListBox** control, based on standard file attributes used in the operating environment. Setting any of these properties with code at run time resets the **FileListBox** control to display only those files with the specified attributes.

For example, in a find-and-replace operation you could display only system files by setting the **System** property to **True** and the other properties to **False**. Or, as part of a file backup procedure, you could set the **Archive** property to **True** to list only those files modified since the previous backup.

AutoRedraw Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAutoRedrawC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAutoRedrawX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAutoRedrawA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAutoRedrawS"}

Returns or sets the output from a graphics method to a persistent graphic.

Syntax

object.**AutoRedraw** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that specifies how the object is repainted, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	Enables automatic repainting of a **Form** object or **PictureBox** control. Graphics and text are written to the screen and to an image stored in memory. The object doesn't receive Paint events; it's repainted when necessary, using the image stored in memory.
False	(Default) Disables automatic repainting of an object and writes graphics or text only to the screen. Visual Basic invokes the object's Paint event when necessary to repaint the object.

Remarks

This property is central to working with the following graphics methods: **Circle**, **Cls**, **Line**, **Point**, **Print**, and **PSet**. Setting **AutoRedraw** to **True** automatically redraws the output from these methods in a **Form** object or **PictureBox** control when, for example, the object is resized or redisplayed after being hidden by another object.

You can set **AutoRedraw** in code at run time to alternate between drawing persistent graphics (such as a background or grid) and temporary graphics. If you set **AutoRedraw** to **False**, previous output becomes part of the background screen. When **AutoRedraw** is set to **False**, background graphics aren't deleted if you clear the drawing area with the **Cls** method. Setting **AutoRedraw** back to **True** and then using **Cls** clears the background graphics.

Note If you set the **BackColor** property, all graphics and text, including the persistent graphic, are erased. In general, all graphics should be displayed using the Paint event unless **AutoRedraw** is set to **True**.

To retrieve the persistent graphic created when **AutoRedraw** is set to **True**, use the **Image** property. To pass the persistent graphic to a Windows API when **AutoRedraw** is set to **True**, use the object's **hDC** property.

If you set a form's **AutoRedraw** property to **False** and then minimize the form, the **ScaleHeight** and **ScaleWidth** properties are set to icon size. When **AutoRedraw** is set to **True**, **ScaleHeight** and **ScaleWidth** remain the size of the restored window.

If **AutoRedraw** is set to **False**, the **Print** method will print on top of graphical controls such as the **Image** and **Shape** controls.

AutoShowChildren Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAutoShowChildrenC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAutoShowChildrenX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAutoShowChildrenA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAutoShowChildrenS"}

Returns or sets a value that determines whether MDI child forms are displayed when loaded.

Syntax

object.**AutoShowChildren** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that specifies whether MDI child forms are automatically visible, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) MDI child forms are automatically displayed when loaded.
False	MDI child forms aren't automatically displayed when loaded.

Remarks

You can use the **AutoShowChildren** property to load MDI child forms and leave them hidden until they're displayed using the **Show** method.

AutoSize Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbProAutosizeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAutoSizeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAutoSizeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAutoSizeS"}

Returns or sets a value that determines whether a control is automatically resized to display its entire contents.

Syntax

object.**AutoSize** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that specifies whether the control is resized, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	Automatically resizes the control to display its entire contents.
False	(Default) Keeps the size of the control constant. Contents are clipped when they exceed the area of the control.

CellSelected Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproCellSelectedC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCellSelectedX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCellSelectedA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCellSelectedS"}

Returns a value that determines whether the cell specified by the **Col** and **Row** properties (the active cell) is within the **Grid** control's selected region.

Syntax

object.**CellSelected**

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

Settings

The settings for **CellSelected** property are:

Setting	Description
True	The specified cell is within the selected area.
False	The specified cell isn't within the selected area.

Remarks

You can use this property to determine whether the user has selected a specific cell or a specific range of cells.

At run time, the user can select a range of cells by clicking a cell and then dragging the mouse or by pressing SHIFT and using the arrow keys.

Use the **SelEndCol**, **SelStartCol**, **SelEndRow**, and **SelStartRow** properties to determine the selected region. Use the **Col** and **Row** properties to determine the active cell.

Clip Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproClipC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproClipX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproClipA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproClipS"}

Returns or sets the contents of the cells in a selected region of a **Grid** control. Not available at design time.

Syntax

object.**Clip** [= *string*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	A string expression containing the cell contents.

Remarks

The *string* can contain the contents of multiple rows and columns. In *string,* a tab character (ANSI character 9) indicates a new cell in a row, and a carriage return (ANSI character 13) indicates the beginning of a new row. Use the **Chr** function to embed these characters in strings. For example, the following line of code puts text into a selected area two rows high and two columns wide:

```
Grid1.Clip = "1st" & Chr(9) & "a" & Chr(13) & "2nd" & Chr(9) & "b"
```

When you put data into a **Grid** control, only the selected cells are affected. If there are more cells in the selected region than are described in *string,* the remaining cells are set to **Null**. If there are more cells described in *string* than in the selected region, the unused portion of *string* is ignored.

ClipControls Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproClipControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproClipControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproClipControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproClipControlsS"}

Returns or sets a value that determines whether graphics methods in Paint events repaint the entire object or only newly exposed areas. Also determines whether the Microsoft Windows operating environment creates a clipping region that excludes nongraphical controls contained by the object. Read-only at run time.

Syntax

object.**ClipControls**

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that specifies how objects are repainted, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) Graphics methods in Paint events repaint the entire object. A clipping region is created around nongraphical controls on the form before a Paint event.
False	Graphics methods in Paint events repaint only newly exposed areas. A clipping region isn't created around nongraphical controls before a Paint event. Complex forms usually load and repaint faster when **ClipControls** is set to **False**.

Remarks

Clipping is the process of determining which parts of a form or container, such as a **Frame** or **PictureBox** control, are painted when the form is displayed. An outline of the form and controls is created in memory. The Windows operating environment uses this outline to paint some parts, such as the background, without affecting other parts, such as the contents of a **TextBox** control. Because the clipping region is created in memory, setting this property to **False** can reduce the time needed to paint or repaint a form.

The clipping region includes most controls, but doesn't clip around the **Image**, **Label**, **Line**, or **Shape** controls.

Avoid nesting intrinsic controls with **ClipControls** set to **True** inside a control with **ClipControls** set to **False** (for instance, a command button inside a picture box). This kind of control nesting causes the controls to repaint incorrectly. To fix this problem, set the **ClipControls** property for both the container control and the nested controls to **True**.

ColAlignment Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproColAlignmentC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproColAlignmentX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproColAlignmentA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproColAlignmentS"}

Returns or sets the alignment of data in a column. Not available at design time.

Syntax

object.**ColAlignment(***column***)** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
column	The number of a column.
number	An integer that specifies the column alignment, as described in Settings.

Settings

The settings for *number* are:

Constant	Setting	Description
vbLeftJustify	0	(Default) Left-aligned.
VbRightJustify	1	Right-aligned.
VbCenter	2	Centered.

Remarks

Any column can have an alignment that is different from other columns. This property doesn't affect cells in a fixed column. To set the alignment for a fixed column, use the **FixedAlignment** property.

ColorMode Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproColorModeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproColorModeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproColorModeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproColorModeS"}

Returns or sets a value that determines whether a color printer prints output in color or monochrome. Not available at design time.

Syntax

object.**ColorMode** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A constant or integer that specifies the print mode, as described in Settings.

Settings

The settings for *value* are:

Setting	Value	Description
vbPRCMMonochrome	1	Print output in monochrome (usually shades of black and white).
VbPRCMColor	2	Print output in color.

Remarks

The default value depends on the printer driver and the current printer settings. Monochrome printers ignore this property.

Note The effect of the properties of the **Printer** object depends on the driver supplied by the printer manufacturer. Some property settings may have no effect, or several different property settings may all have the same effect. If you set the **ColorMode** property for a printer which doesn't support color, the setting is ignored. If you attempt to reference the **ColorMode** property, however, you will get an error message. Settings outside the accepted range may also produce an error. For more information, see the manufacturer's documentation for the specific driver.

Columns Property (ListBox)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproColumnsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproColumnsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproColumnsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproColumnsS"}

Returns or sets a value that determines whether a **ListBox** control scrolls vertically or horizontally and how the items in the columns are displayed. If it scrolls horizontally, the **Columns** property determines how many columns are displayed.

Syntax

object.**Columns** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	An integer that specifies how a control scrolls and how items are arranged in columns, as described in Settings.

Settings

The settings for *number* are:

Setting	Description
0	(Default) Items are arranged in a single column and the **ListBox** scrolls vertically.
1 to *n*	Items are arranged in snaking columns, filling the first column, then the second column, and so on. The **ListBox** scrolls horizontally and displays the specified number of columns.

Remarks

For horizontal-scrolling **ListBox** controls, the column width is equal to the width of the **ListBox** divided by the number of columns.

This property can't be set to 0 or changed from 0 at run time — that is, you can't change a multiple-column **ListBox** to a single-column **ListBox** or a single-column **ListBox** to a multiple-column **ListBox** at run time. However, you can change the number of columns in a multiple-column **ListBox** at run time.

ColWidth Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproColWidthC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproColWidthX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproColWidthA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproColWidthS"}

Returns or sets the width of the specified column in twips. Not available at design time.

Syntax

object.**ColWidth(***column***)** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
column	The number of a column.
number	A numeric expression that specifies the column width.

Remarks

You can use this property to set the width of any column at run time. Users can also change the width of a column by positioning the mouse pointer between columns and then dragging.

Copies Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproCopiesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCopiesX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCopiesA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCopiesS"}

Returns or sets a value that determines the number of copies to be printed. For the **Printer** object, not available at design time.

Syntax

object.**Copies** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression that specifies the number of copies to print. This value must be an integer.

Remarks

For the Print dialog box, this property returns the number of copies entered by the user in the Copies box. If the **cdlPDUseDevModeCopies** flag is set for the **CommonDialog** control, this property always returns 1.

For the **Printer** object, multiple copies may or may not be collated, depending on the printer driver. Multiple copies of the entire document or multiple copies of each page may be printed. For printers that don't support collating, set **Copies** = 1, and then use a loop in code to print multiple copies of the entire document.

Note The effect of the properties of the **Printer** object depends on the driver supplied by the printer manufacturer. Some property settings may have no effect, or several different property settings may all have the same effect. Settings outside the accepted range may produce an error. For more information, see the manufacturer's documentation for the specific driver.

CurrentX, CurrentY Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproCurrentXC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCurrentXX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCurrentXA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCurrentXS"}

Return or set the horizontal (**CurrentX**) or vertical (**CurrentY**) coordinates for the next printing or drawing method. Not available at design time.

Syntax

object.**CurrentX** [= *x*]
object.**CurrentY** [= *y*]

The **CurrentX** and **CurrentY** properties syntax have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
x	A number that specifies the horizontal coordinate.
y	A number that specifies the vertical coordinate.

Remarks

Coordinates are measured from the upper-left corner of an object. The **CurrentX** property setting is 0 at an object's left edge, and the **CurrentY** property setting is 0 at its top edge. Coordinates are expressed in twips, or the current unit of measurement defined by the **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, **ScaleTop**, and **ScaleMode** properties.

When you use the following graphics methods, the **CurrentX** and **CurrentY** settings are changed as indicated:

This method	Sets CurrentX, CurrentY to
Circle	The center of the object.
Cls	0, 0.
EndDoc	0, 0.
Line	The end point of the line.
NewPage	0, 0.
Print	The next print position.
PSet	The point drawn.

Default Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDefaultC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDefaultX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDefaultA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDefaultS"}

Returns or sets a value that determines which **CommandButton** control is the default command button on a form.

Syntax

object.**Default** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that specifies whether the command button is the default, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The **CommandButton** is the default command button.
False	(Default) The **CommandButton** isn't the default command button.

Remarks

Only one command button on a form can be the default command button. When **Default** is set to **True** for one command button, it's automatically set to **False** for all other command buttons on the form. When the command button's **Default** property setting is **True** and its parent form is active, the user can choose the command button (invoking its Click event) by pressing ENTER. Any other control with the focus doesn't receive a keyboard event (KeyDown, KeyPress, or KeyUp) for the ENTER key unless the user has moved the focus to another command button on the same form. In this case, pressing ENTER chooses the command button that has the focus instead of the default command button.

For a form or dialog box that supports an irreversible action such as a delete operation, make the Cancel button the default command button by setting its **Default** property to **True**.

For **OLE** container controls, the **Default** property is provided only for those objects that specifically behave like **CommandButton** controls.

DeviceName Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDeviceNameC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDeviceNameX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDeviceNameA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDeviceNameS"}

Returns the name of the device a driver supports.

Syntax

object.**DeviceName**

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

Remarks

Each printer driver supports one or more devices — for example, HP LaserJet IIISi is a device name.

Note The effect of properties of the **Printer** object depends on the driver supplied by the printer manufacturer. Some property settings may have no effect, or several different property settings may all have the same effect. Settings outside the accepted range may produce an error. For more information, see the manufacturer's documentation for the specific driver.

DragIcon Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDragIconC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDragIconX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDragIconA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDragIconS"}

Returns or sets the icon to be displayed as the pointer in a drag-and-drop operation.

Syntax

object.**DragIcon** [= *icon*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
icon	Any code reference that returns a valid icon, such as a reference to a form's icon (`Form1.Icon`), a reference to another control's **DragIcon** property (`Text1.DragIcon`), or the **LoadPicture** function.

Settings

The settings for *icon* are:

Setting	Description
(none)	(Default) An arrow pointer inside a rectangle.
Icon	A custom mouse pointer. You specify the icon by setting it using the Properties window at design time. You can also use the **LoadPicture** function at run time. The file you load must have the .ico filename extension and format.

Remarks

You can use the **DragIcon** property to provide visual feedback during a drag-and-drop operation — for example, to indicate that the source control is over an appropriate target. **DragIcon** takes effect when the user initiates a drag-and-drop operation. Typically, you set **DragIcon** as part of a MouseDown or DragOver event procedure.

Note At run time, the **DragIcon** property can be set to any object's **DragIcon** or **Icon** property, or you can assign it an icon returned by the **LoadPicture** function.

When you set the **DragIcon** property at run time by assigning the **Picture** property of one control to the **DragIcon** property of another control, the **Picture** property must contain an .ico file, not a .bmp file.

DragMode Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDragModeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDragModeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDragModeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDragModeS"}

Returns or sets a value that determines whether manual or automatic drag mode is used for a drag-and-drop operation.

Syntax

object.**DragMode** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	An integer that specifies the drag mode, as described in Settings.

Settings

The settings for *number* are:

Constant	Setting	Description
vbManual	0	(Default) Manual — requires using the **Drag** method to initiate a drag-and-drop operation on the source control.
vbAutomatic	1	Automatic — clicking the source control automatically initiates a drag-and-drop operation. **OLE** container controls are automatically dragged only when they don't have the focus.

Remarks

When **DragMode** is set to 1 (Automatic), the control doesn't respond as usual to mouse events. Use the 0 (Manual) setting to determine when a drag-and-drop operation begins or ends; you can use this setting to initiate a drag-and-drop operation in response to a keyboard or menu command or to enable a source control to recognize a MouseDown event prior to a drag-and-drop operation.

Clicking while the mouse pointer is over a target object or form during a drag-and-drop operation generates a DragDrop event for the target object. This ends the drag-and-drop operation. A drag-and-drop operation may also generate a DragOver event.

Note While a control is being dragged, it can't recognize other user-initiated mouse or keyboard events (KeyDown, KeyPress or KeyUp, MouseDown, MouseMove, or MouseUp). However, the control can receive events initiated by code or by a DDE link.

DrawMode Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDrawModeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDrawModeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDrawModeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDrawModeS"}

Returns or sets a value that determines the appearance of output from graphics method or the appearance of a **Shape** or **Line** control.

Syntax

object.**DrawMode** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	An integer that specifies appearance, as described in Settings.

Settings

The settings for *number* are:

Constant	Setting	Description
vbBlackness	1	Blackness.
vbNotMergePen	2	Not Merge Pen — Inverse of setting 15 (Merge Pen).
vbMaskNotPen	3	Mask Not Pen — Combination of the colors common to the background color and the inverse of the pen.
vbNotCopyPen	4	Not Copy Pen — Inverse of setting 13 (Copy Pen).
vbMaskPenNot	5	Mask Pen Not — Combination of the colors common to both the pen and the inverse of the display.
vbInvert	6	Invert — Inverse of the display color.
vbXorPen	7	Xor Pen — Combination of the colors in the pen and in the display color, but not in both.
vbNotMaskPen	8	Not Mask Pen — Inverse of setting 9 (Mask Pen).
vbMaskPen	9	Mask Pen — Combination of the colors common to both the pen and the display.
vbNotXorPen	10	Not Xor Pen — Inverse of setting 7 (Xor Pen).
vbNop	11	Nop — No operation — output remains unchanged. In effect, this setting turns drawing off.
vbMergeNotPen	12	Merge Not Pen — Combination of the display color and the inverse of the pen color.
vbCopyPen	13	Copy Pen (Default) — Color specified by the **ForeColor** property.
vbMergePenNot	14	Merge Pen Not — Combination of the pen

		color and the inverse of the display color.
vbMergePen	15	Merge Pen — Combination of the pen color and the display color.
vbWhiteness	16	Whiteness.

Remarks

Use this property to produce visual effects with **Shape** or **Line** controls or when drawing with the graphics methods. Visual Basic compares each pixel in the draw pattern to the corresponding pixel in the existing background and then applies bit-wise operations. For example, setting 7 (Xor Pen) uses the **Xor** operator to combine a draw pattern pixel with a background pixel.

The exact effect of a **DrawMode** setting depends on the way the color of a line drawn at run time combines with colors already on the screen. Settings 1, 6, 7, 11, 13, and 16 yield the most predictable results.

DrawStyle Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDrawStyleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDrawStyleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDrawStyleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDrawStyleS"}

Returns or sets a value that determines the line style for output from graphics methods.

Syntax

object.**DrawStyle** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	An integer that specifies line style, as described in Settings.

Settings

The settings for *number* are:

Constant	Setting	Description
vbSolid	0	(Default) Solid
vbDash	1	Dash
vbDot	2	Dot
vbDashDot	3	Dash-Dot
vbDashDotDot	4	Dash-Dot-Dot
vbInvisible	5	Transparent
vbInsideSolid	6	Inside Solid

Remarks

If **DrawWidth** is set to a value greater than 1, **DrawStyle** settings 1 through 4 produce a solid line (the **DrawStyle** property value isn't changed). If **DrawWidth** is set to 1, **DrawStyle** produces the effect described in the preceding table for each setting.

Drive Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDriveC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDriveX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDriveA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDriveS"}

Returns or sets the selected drive at run time. Not available at design time.

Syntax

object.**Drive** [= *drive*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
drive	A string expression that specifies the selected drive.

Remarks

The valid drives for the **Drive** property include all drives present in or connected to the system when the control is created or refreshed at run time. The default setting of the **Drive** property is the current drive.

When reading this property setting, the selected drive is returned in one of the following formats:

- Floppy disks — "a:" or "b:", and so on
- Fixed media — "c: [*volume id*]"
- Network connections — "x: \\server\share"

When setting this property:

- Only the first character of the string is significant (the string isn't case-sensitive).
- Changing the setting for the **Drive** property invokes a Change event.
- Selecting a drive that isn't present causes an error.
- Setting this property also regenerates the drive list, providing a way in code to track network connections added since the control was created.

If the **FileName** property is set to a qualified network path without a drive designation, the value of the **Drive** property is a zero-length string (""), no drive is selected, and the **ListIndex** property setting is 1.

Note The **Drive** property returns a different value from the **ListIndex** property, which returns the list box selection.

DriverName Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDriverNameC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDriverNameX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDriverNameA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDriverNameS"}

Returns the name of the driver for a **Printer** object.

Syntax

object.**DriverName**

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

Remarks

Each driver has a unique name. For example, the **DriverName** for several of the Hewlett-Packard printers is HPPCL5MS. The **DriverName** is typically the driver's filename without an extension.

Note The effect of the properties of the **Printer** object depends on the driver supplied by the printer manufacturer. Some property settings may have no effect, or several different property settings may all have the same effect. Settings outside the accepted range may produce an error. For more information, see the manufacturer's documentation for the specific driver.

Duplex Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDuplexC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDuplexX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDuplexA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDuplexS"}

Returns or sets a value that determines whether a page is printed on both sides (if the printer supports this feature). Not available at design time.

Syntax

object.**Duplex** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A value or constant that specifies the type of printing, as described in Settings.

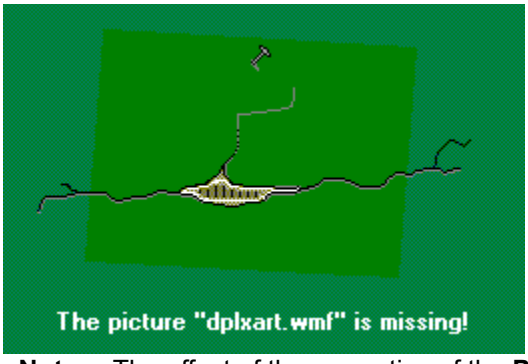
Settings

The settings for *value* are:

Constant	Value	Description
vbPRDPSimplex	1	Single-sided printing with the current orientation setting.
VbPRDPHorizontal	2	Double-sided printing using a horizontal page turn.
VbPRDPVertical	3	Double-sided printing using a vertical page turn.

Remarks

With horizontal duplex printing, the top of both sides of the page are at the same end of the sheet. With vertical duplex printing, the bottom of one page is at the same end of the sheet as the top of the next page. The following diagram illustrates horizontal and vertical duplex printing:



Note The effect of the properties of the **Printer** object depends on the driver supplied by the printer manufacturer. Some property settings may have no effect, or several different property settings may all have the same effect. Settings outside the accepted range may produce an error. For more information, see the manufacturer's documentation for the specific driver.

FileName Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFileNameC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFileNameX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFileNameA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFileNames"}

Returns or sets the path and filename of a selected file. Not available at design time for the **FileListBox** control and **ProjectTemplate** object.

Syntax

object.**FileName** [= *pathname*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
pathname	A string expression that specifies the path and filename.

Remarks

When you create the control at run time, the **FileName** property is set to a zero-length string (""), meaning no file is currently selected.

In the **CommonDialog** control, you can set the **FileName** property before opening a dialog box to set an initial filename.

Reading this property returns the currently selected filename from the list. The path is retrieved separately, using the **Path** property. The value is functionally equivalent to **List**(*ListIndex*). If no file is selected, **FileName** returns a zero-length string.

When setting this property:

- Including a drive, path, or pattern in the string changes the settings of the **Drive**, **Path**, and **Pattern** properties accordingly.
- Including the name of an existing file (without wildcard characters) in the string selects the file.
- Changing the value of this property may also cause one or more of these events: PathChange (if you change the path), PatternChange (if you change the pattern), or DblClick (if you assign an existing filename).
- This property setting can be a qualified network path and filename using the following syntax:
  `\\servername\sharename\pathname`

FillColor Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFillColorC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFillColorX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFillColorA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFillColorS"}

Returns or sets the color used to fill in shapes; **FillColor** is also used to fill in circles and boxes created with the **Circle** and **Line** graphics methods.

Syntax

object.**FillColor** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A value or constant that determines the fill color, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
Normal RGB colors	Colors set with the **RGB** or **QBColor** functions in code.
System default colors	Colors specified with the system color constants in the Visual Basic (VB) object library in the Object Browser. The Microsoft Windows operating environment substitutes the user's choices, as specified by the user's Control Panel settings.

By default, **FillColor** is set to 0 (Black).

Remarks

Except for the **Form** object, when the **FillStyle** property is set to its default, 1 (Transparent), the **FillColor** setting is ignored.

FillStyle Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFillStyleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFillStyleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFillStyleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFillStyleS"}

- **Grid** control — Returns or sets a value that determines whether setting the **Text** property of a **Grid** causes the value to be displayed in all selected cells.
- All other objects — Returns or sets the pattern used to fill **Shape** controls as well as circles and boxes created with the **Circle** and **Line** graphics methods.

Syntax

object.**FillStyle** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	An integer that specifies the fill style, as described in Settings.

Settings

The *number* settings for a **Grid** control are:

Constant	Setting	Description
grdSingle	0	Single — Changing the **Text** property affects only the active cell.
grdRepeat	1	Repeat — Changing the **Text** property affects all selected cells.

The *number* settings for other objects are:

Constant	Setting	Description
vbFSSolid	0	Solid
vbFSTransparent	1	(Default) Transparent
vbHorizontalLine	2	Horizontal Line
vbVerticalLine	3	Vertical Line
vbUpwardDiagonal	4	Upward Diagonal
vbDownwardDiagonal	5	Downward Diagonal
vbCross	6	Cross
vbDiagonalCross	7	Diagonal Cross

Remarks

When **FillStyle** is set to 1 (Transparent), the **FillColor** property is ignored, except for the **Form** object.

FixedAlignment Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFixedAlignmentC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFixedAlignmentX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFixedAlignmentA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFixedAlignmentS"}

Returns or sets a value that determines the alignment of data in the fixed cells of a column. Not available at design time.

Syntax

object.**FixedAlignment(***column***)** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
column	The number of a column.
number	An integer that specifies alignment, as described in Settings.

Settings

The settings for *number* are:

Constant	Setting	Description
grdAlignLeft	0	(Default) Left-aligned
grdAlignRight	1	Right-aligned
grdAlignCenter	2	Centered
	3	Use the **ColAlignment** setting for the column specified by *column*

Remarks

The **FixedAlignment** property behaves like the **ColAlignment** property except that it only affects the alignment of fixed cells. You can use **FixedAlignment** to align headings differently from the rest of the columns.

FixedCols, FixedRows Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFixedColsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFixedColsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFixedColsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFixedColsS"}

Return or set the total number of fixed columns or fixed rows for a **Grid** control. By default, a **Grid** has one fixed column and one fixed row.

Syntax

object.**FixedCols** [= *number*]
object.**FixedRows** [= *number*]

The **FixedCols** and **FixedRows** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression that specifies the number of fixed columns or rows.

Remarks

A fixed column is a stationary column on the left side of the **Grid** control. A fixed row is a stationary row along the top of the **Grid**. You can have zero or more fixed columns and zero or more fixed rows. Fixed columns and rows are displayed in gray and don't move when the other columns or rows in the **Grid** are scrolled. You can't change the color of a fixed column or row.

Fixed columns and rows are typically used in spreadsheet applications to display row numbers and column names or letters.

A **Grid** control must have at least one nonfixed column and one nonfixed row. The maximum number of fixed columns or rows allowed in a **Grid** is one less than the total number of columns or rows.

FontCount Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFontCountC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontCountX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontCountA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontCountS"}

Returns the number of fonts available for the current display device or active printer.

Syntax

object.**FontCount**

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

Remarks

Use this property with the **Fonts** property to see a list of available screen or printer fonts. Fonts available in Visual Basic vary according to your system configuration, display devices, and printing devices.

Fonts Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFontsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontsS"}

Returns all font names available for the current display device or active printer.

Syntax

object.**Fonts(***index***)**

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer from 0 to **FontCount** 1.

Remarks

The **Fonts** property works in conjunction with the **FontCount** property, which returns the number of font names available for the object. Fonts available in Visual Basic vary according to your system configuration, display devices, and printing devices. Use both the **Fonts** and the **FontCount** properties to get information about available screen or printer fonts.

GridLines Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproGridLinesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproGridLinesX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproGridLinesA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproGridLinesS"}

Returns or sets a value that determines whether the lines between cells are visible.

Syntax

object.**Gridlines** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A value that specifies the visibility of grid lines, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) Grid lines are visible
False	Grid lines aren't visible

hDC Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbprohDCC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbprohDCX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHDCA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbprohDCS"}

Returns a handle provided by the Microsoft Windows operating environment to the device context of an object.

Syntax

object.**hDC**

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

Remarks

This property is a Windows operating environment device context handle. The Windows operating environment manages the system display by assigning a device context for the **Printer** object and for each form and **PictureBox** control in your application. You can use the **hDC** property to refer to the handle for an object's device context. This provides a value to pass to Windows API calls.

With a **CommonDialog** control, this property returns a device context for the printer selected in the Print dialog box when the **cdlReturnDC** flag is set or an information context when the **cdlReturnIC** flag is set.

Note The value of the **hDC** property can change while a program is running, so don't store the value in a variable; instead, use the **hDC** property each time you need it.

The **AutoRedraw** property can cause the **hDC** property setting to change. If **AutoRedraw** is set to **True** for a form or **PictureBox** container, **hDC** acts as a handle to the device context of the persistent graphic (equivalent to the **Image** property). When **AutoRedraw** is **False**, **hDC** is the actual **hDC** value of the Form window or the **PictureBox** container. The **hDC** property setting may change while the program is running regardless of the **AutoRedraw** setting.

HideSelection Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHideSelectionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHideSelectionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHideSelectionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHideSelectionS"}

Returns a value that determines whether selected text appears highlighted when a control loses the focus.

Syntax

object.**HideSelection**

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

Return Values

The **HideSelection** property return values are:

Value	Description
True	(Default) Selected text doesn't appear highlighted when the control loses the focus.
False	Selected text appears highlighted when the control loses the focus.

Remarks

You can use this property to indicate which text is highlighted while another form or a dialog box has the focus — for example, in a spell-checking routine.

HighLight Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHighLightC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHighLightX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHighLightA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHighLightS"}

Returns or sets a value that determines whether selected cells appear highlighted.

Syntax

object.**Highlight** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A value that specifies selected cells' appearance, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) Selected cells are highlighted.
False	Selected cells aren't highlighted.

Remarks

When this property is set to **False** and the user selects a range of cells, there is no visual cue that shows which cells are selected.

hWnd Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbprohWndC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbprohWndX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbprohWndA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbprohWndS"}

Returns a handle to a form or control.

Note This property is not supported for the **OLE** container control.

Syntax

object.**hWnd**

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

Remarks

The Microsoft Windows operating environment identifies each form and control in an application by assigning it a handle, or **hWnd**. The **hWnd** property is used with Windows API calls. Many Windows operating environment functions require the **hWnd** of the active window as an argument.

Note Because the value of this property can change while a program is running, never store the **hWnd** value in a variable.

Image Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproImageC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproImageX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproImageA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproImageS"}

Returns a handle to a persistent graphic; the handle is provided by the Microsoft Windows operating environment.

Syntax

object.**Image**

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

Remarks

An object's **AutoRedraw** property determines whether the repainting of an object occurs with a persistent graphics or through Paint events. The Windows operating environment identifies an object's persistent graphic by assigning a handle to it; you can use the **Image** property to get this handle.

An **Image** value exists regardless of the setting for the **AutoRedraw** property. If **AutoRedraw** is **True** and nothing has been drawn, the image displays only the color set by the **BackColor** property and the picture.

You can assign the value of **Image** to the **Picture** property. The **Image** property also provides a value to pass to Windows API calls.

The **Image**, **DragIcon**, and **Picture** properties are normally used when assigning values to other properties, when saving with the **SavePicture** statement, or when placing something on the Clipboard. You can't assign these to a temporary variable, other than the Picture data type.

The **AutoRedraw** property can cause **Image**, which is a handle to a bitmap, to change. When **AutoRedraw** is **True**, an object's **hDC** property becomes a handle to a device context that contains the bitmap returned by **Image**.

ItemData Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproItemDataC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproItemDataX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproItemDataA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproItemDataS"}

Returns or sets a specific number for each item in a **ComboBox** or **ListBox** control.

Syntax

object.**ItemData(***index***)** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	The number of a specific item in the object.
number	The number to be associated with the specified item.

Remarks

The **ItemData** property is an array of long integer values with the same number of items as a control's **List** property. You can use the numbers associated with each item to identify the items. For example, you can use an employee's identification number to identify each employee name in a **ListBox** control. When you fill the **ListBox**, also fill the corresponding elements in the **ItemData** array with the employee numbers.

The **ItemData** property is often used as an index for an array of data structures associated with items in a **ListBox** control.

Note When you insert an item into a list with the **AddItem** method, an item is automatically inserted in the **ItemData** array as well. However, the value isn't reinitialized to zero; it retains the value that was in that position before you added the item to the list. When you use the **ItemData** property, be sure to set its value when adding new items to a list.

KeyPreview Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproKeyPreviewC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproKeyPreviewX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproKeyPreviewA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproKeyPreviewS"}

Returns or sets a value that determines whether keyboard events for forms are invoked before keyboard events for controls. The keyboard events are KeyDown, KeyUp, and KeyPress.

Syntax

object.**KeyPreview** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that specifies how events are received, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The form receives keyboard events first and then the active control.
False	(Default) The active control receives keyboard events; the form doesn't.

Remarks

You can use this property to create a keyboard-handling procedure for a form. For example, when an application uses function keys, you'll want to process the keystrokes at the form level rather than writing code for each control that might receive keystroke events.

If a form has no visible and enabled controls, it automatically receives all keyboard events.

To handle keyboard events only at the form level and not allow controls to receive keyboard events, set *KeyAscii* to 0 in the form's KeyPress event, and set *KeyCode* to 0 in the form's KeyDown event.

Note Some controls intercept keyboard events so that the form can't receive them. Examples include the ENTER key when focus is on a **CommandButton** control and arrow keys when focus is on a **ListBox** control.

LargeChange, SmallChange Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLargeChangeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLargeChangeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLargeChangeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLargeChangeS"}

- **LargeChange** — Returns or sets the amount of change to the **Value** property setting in a scroll bar control (**HScrollBar** or **VScrollBar**) when the user clicks the area between the scroll box and scroll arrow.
- **SmallChange** — Returns or sets the amount of change to the **Value** property setting in a scroll bar control when the user clicks a scroll arrow.

Syntax

object.**LargeChange** [= *number*]
object.**SmallChange** [= *number*]

The **LargeChange** and **SmallChange** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	An integer that specifies the amount of change to the **Value** property.

Remarks

For both properties, you can specify an integer between 1 and 32,767, inclusive. By default, each property is set to 1.

The Microsoft Windows operating environment automatically sets proportional scrolling increments for scroll bars on **MDI Form** objects, **ComboBox** controls, and **ListBox** controls based on the amount of data in the object. For the **HScrollBar** and **VScrollBar** controls, however, you must specify these increments. Use **LargeChange** and **SmallChange** to set scrolling increments appropriate to how the scroll bar is being used.

Typically, you set **LargeChange** and **SmallChange** at design time. You can also reset them in code at run time when the scrolling increment must change dynamically.

Note You set the maximum and minimum ranges of the **HScrollBar** and **VScrollBar** controls with the **Max** and **Min** properties.

LeftCol Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLeftColC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLeftColX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLeftColA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLeftColS"}

Returns or sets the leftmost visible column (other than a fixed column) in a grid. Not available at design time.

Syntax

object.**LeftCol** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression that specifies the number of the leftmost visible column. Default is 0.

Remarks

You can use the **LeftCol** property in code to scroll a **Grid** or **DBGrid** control programmatically. Use the **TopRow** property to determine the topmost visible row in the **Grid** or **DBGrid**.

LinkItem Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLinkItemC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLinkTopicX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLinkItemA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLinkItemS"}

Returns or sets the data passed to a destination control in a DDE conversation with another application.

Syntax

object.**LinkItem** [= *string*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	A string expression that specifies the data to be passed to the destination control.

Remarks

This property corresponds to the *item* argument in the standard DDE syntax, with *application*, *topic*, and *item* as arguments. To set this property, specify a recognizable unit of data in an application as a reference — for example, a cell reference such as "R1C1" in Microsoft Excel.

Use **LinkItem** in combination with the **LinkTopic** property to specify the complete data link for a destination control to a source application. To activate this link, set the **LinkMode** property.

You set **LinkItem** only for a control used as a destination. When a Visual Basic form is a source in a DDE conversation, the name of any **Label**, **PictureBox**, or **TextBox** control on the form can be the *item* argument in the *application*|*topic*!*item* string used by the destination. For example, the following syntax represents a valid reference from Microsoft Excel to a Visual Basic application:

=*VizBasicApplication*|*MyForm*!*TextBox1*

You could enter the preceding syntax for a destination cell in the Microsoft Excel formula bar.

A DDE control can potentially act as destination and source simultaneously, causing an infinite loop if a destination-source pair is also a source-destination pair with itself. For instance, a **TextBox** control may be both a source (through its parent form) and destination of the same cell in Microsoft Excel. When data in a Visual Basic **TextBox** changes, sending data to Microsoft Excel, the cell in Microsoft Excel changes, sending the change to the **TextBox**, and so on, causing the loop.

To avoid such loops, use related but not identical items for destination-source and source-destination links in both directions between applications. For example, in Microsoft Excel, use related cells (precedents or dependents) to link a worksheet with a Visual Basic control, avoiding use of a single item as both destination and source. Document any *application*|*topic* pairs you establish if you include a Paste Link command for run-time use.

Note Setting a permanent data link at design time with the Paste Link command from the Edit menu also sets the **LinkMode**, **LinkTopic**, and **LinkItem** properties. This creates a link that is saved with the form. Each time the form is loaded, Visual Basic attempts to re-establish the conversation.

LinkMode Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLinkModeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLinkTopicX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLinkModeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLinkModeS"}

Returns or sets the type of link used for a DDEconversation and activates the connection as follows:

- Control — Allows a destination control on a Visual Basic form to initiate a conversation, as specified by the control's **LinkTopic** and **LinkItem** properties.
- Form — Allows a destination application to initiate a conversation with a Visual Basic source form, as specified by the destination application's *application***|***topic***!***item* expression.

Syntax

object.**LinkMode** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	An integer that specifies the type of connection, as described in Settings.

Settings

For controls used as destinations in DDE conversations, the settings for *number* are:

Constant	Setting	Description
vbLinkNone	0	(Default) None — No DDE interaction.
vbLinkAutomatic	1	Automatic — Destination control is updated each time the linked data changes.
vbLinkManual	2	Manual — {Destination control is updated only when the **LinkRequest** method is invoked.
vbLinkNotify	3	Notify — A LinkNotify event occurs whenever the linked data changes, but the destination control is updated only when the **LinkRequest** method is invoked.

For forms used as sources in DDE conversations, the settings for *number* are:

Constant	Setting	Description
vbLinkNone	0	(Default) None — No DDE interaction. No destination application can initiate a conversation with the source form as the topic, and no application can poke data to the form. If **LinkMode** is 0 (None) at design time, you can't change it to 1 (Source) at run time.
vbLinkSource	1	Source — Allows any **Label**, **PictureBox**, or **TextBox** control on a form to supply data to any destination application that establishes a DDE conversation with the form. If such a link exists, Visual Basic

automatically notifies the destination whenever the contents of a control are changed. In addition, a destination application can poke data to any **Label**, **PictureBox**, or **TextBox** control on the form. If **LinkMode** is 1 (Source) at design time, you can change it to 0 (None) and back at run time.

Remarks

The following conditions also apply to the **LinkMode** property:

- Setting **LinkMode** to a nonzero value for a destination control causes Visual Basic to attempt to initiate the conversation specified in the **LinkTopic** and **LinkItem** properties. The source updates the destination control according to the type of link specified (automatic, manual, or notify).
- If a source application terminates a conversation with a Visual Basic destination control, the value for that control's **LinkMode** setting changes to 0 (None).
- If you leave **LinkMode** for a form set to the default 0 (None) at design time, you can't change **LinkMode** at run time. If you want a form to act as a source, you must set **LinkMode** to 1 (Source) at design time. You can then change the value of **LinkMode** at run time.

Note Setting a permanent data link at design time with the Paste Link command from the Edit menu also sets the **LinkMode**, **LinkTopic**, and **LinkItem** properties. This creates a link that is saved with the form. Each time the form is loaded, Visual Basic attempts to re-establish the conversation.

LinkTimeout Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLinkTimeoutC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLinkTimeoutX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLinkTimeoutA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLinkTimeoutS"}

Returns or sets the amount of time a control waits for a response to a DDE message.

Syntax

object.**LinkTimeout** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression that specifies the wait time.

Remarks

By default, the **LinkTimeout** property is set to 50 (equivalent to 5 seconds). You can specify other settings in tenths of a second.

DDE response time from source applications varies. Use this property to adjust the time a destination control waits for a response from a source application. If you use **LinkTimeout**, you can avoid generating a Visual Basic error if a given source application takes too long to respond.

Note The maximum length of time that a control can wait is 65,535 tenths of a second, or about 1 hour 49 minutes. Setting **LinkTimeout** to 1 tells the control to wait the maximum length of time for a response in a DDE conversation. The user can force the control to stop waiting by pressing the ESC key.

Locked Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproReadOnlyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproReadOnlyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproReadOnlyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLockedS"}

Returns or sets a value indicating whether a control can be edited.

Syntax

object.**Locked** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that specifies whether the control can be edited, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	**TextBox** control — you can scroll and highlight the text in the control, but you can't edit it. The program can still modify the text by changing the **Text** property.
	Column object — you can't edit the values in the column.
	ComboBox object — you can't type in the textbox or drop down its list.
False	**TextBox** control — you can edit the text in the control.
	Column object — you can edit the values in the column.
	ComboBox object — you can type in the textbox and drop down its list.

Remarks

For the **Column** object, the default setting of **Locked** is the value of the **DataUpdatable** property for the underlying field; however, if **Column** is unbound or the data source doesn't support **DataUpdatable**, the default is **True**. If **DataUpdatable** in the underlying field is **False**, you do not create an error by setting this property to **True**. However, an error will occur when the control attempts to write the changed data to the database.

For the **ComboBox** control, when **Locked** is set to **True**, the user cannot change any data, but can highlight data in the text box and copy it. This property does not affect programmatic access to the **ComboBox**.

ReadOnly Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproReadC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproReadX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproReadA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":""}

Returns or sets a value that determines whether a **FileListBox** control contains files with read-only attributes.

Syntax

object.**ReadOnly** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that specifies whether the control displays files with read-only attributes, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The control includes read-only files in the list.
False	No read-only files are listed in the control.

Remarks

Use the **ReadOnly** property with a **FileListBox** control to specify whether files with read-only attributes are displayed in the file list or not.

Align Property Example

This example uses a **PictureBox** control as a toolbar on an **MDIForm** object, with a **CommandButton** control to move the **PictureBox** from the top to the bottom of the form. To try this example, create a new **MDIForm** and set the **MDIChild** property of Form1 to **True**. Draw a **PictureBox** on the **MDIForm**, and put a **CommandButton** on the **PictureBox**. Paste the code into the Declarations section of the **MDIForm**, and then press F5. Click the **CommandButton** to move the **PictureBox**.

```
Private Sub Command1_Click ()
   If Picture1.Align = vbAlignTop Then
      Picture1.Align = vbAlignBottom
      ' Align to bottom of form.
   Else
      Picture1.Align = vbAlignTop
      ' Align to top of form.
   End If
End Sub
```

AutoRedraw Property Example

This example alternately displays two graphics on a **PictureBox** control: a persistent filled circle and temporary vertical lines. Click the **PictureBox** to draw or redraw the lines. Resizing the form requires the temporary graphic to be redrawn. To try this example, paste the code into the Declarations section of a form that has a **PictureBox** control named Picture1. Press F5 to run the program, and click the graphic each time you resize the form.

```
Private Sub Form_Load ()
   Picture1.ScaleHeight = 100      ' Set scale to 100.
   Picture1.ScaleWidth = 100
   Picture1.AutoRedraw = True      ' Turn on AutoRedraw.
   Picture1.ForeColor = 0  ' Set ForeColor.
   Picture1.FillColor = QBColor(9) ' Set FillColor.
   Picture1.FillStyle = 0  ' Set FillStyle.
   Picture1.Circle (50, 50), 30    ' Draw a circle.
   Picture1.AutoRedraw = False     ' Turn off AutoRedraw.
End Sub

Private Sub Picture1_Click ()
   Dim I ' Declare variable.
   Picture1.ForeColor = Rgb(Rnd * 255, 0, 0)    ' Select random color.
   For I = 5 To 95 Step 10 ' Draw lines.
      Picture1.Line (I, 0)-(I, 100)
   Next
End Sub
```

AutoShowChildren Property Example

This example presents an **MDIForm** object with an MDI child form, uses the **AutoShowChildren** property to create a hidden form as another instance of the MDI child form, and then creates a visible MDI child form. To try this example, set the **MDIChild** property to **True** on Form1, and then create an **MDIForm** with the MDI Form command on the Insert menu. Copy the code into the Declarations section of the **MDIForm**, and then press F5 to run the program.

```
Private Sub MDIForm_Load()
   MDIForm1.AutoShowChildren = False     ' Set to hide child forms.
   Dim HideForm As New Form1 ' Declare new form.
   HideForm.Caption = "HideForm"   ' Set its caption.
   Load HideForm  ' Load it; it's hidden.
   MDIForm1.AutoShowChildren = True      ' Set to show child forms.
   Dim ShowForm As New Form1 ' Declare another new form.
   ShowForm.Caption = "ShowForm"   ' Set its caption.
   Load ShowForm  ' Load it; it's displayed.
End Sub
```

CellSelected Example

This example selects a cell at random and displays a message that tells you whether it's selected or not when the **Grid** control is clicked. To try this example, use the Components dialog box to add a **Grid** control to the toolbox (from the Project menu, choose Components, and then select Microsoft Grid Control), paste the code into the Declarations section of a new form, and then draw a **Grid** control and a **Label** control. For the **Label**, set the **AutoSize** property to **True**. Press F5 to run the program.

```
Dim TRow As Integer
Dim TCol As Integer

Private Sub Form_Load ()
   Grid1.Rows = 8 ' Set rows and columns.
   Grid1.Cols = 5
   Randomize   ' Choose a cell at random.
   TRow = Int(Rnd * (Grid1.Rows - 1)) + 1
   TCol = Int(Rnd * (Grid1.Cols - 1)) + 1
End Sub

Private Sub Grid1_Click ()
   Grid1.Row = TRow
   Grid1.Col = TCol
' Display message indicating whether clicked cell is selected.
   If Grid1.CellSelected Then      ' If clicked cell selected,
      Label1.Caption = "You found it!"   ' message displayed.
   Else
      Label1.Caption = "Not yet. Try again."
   End If
End Sub
```

Clip Property Example

This example displays the current date and time in the selected area of the **Grid** control when the user clicks the form. To try this example, use the Components dialog box to add a **Grid** control to the toolbox (from the Project menu, choose OLE Components, and then select Microsoft Grid Control), paste the code into the Declarations section of a new form, and then draw a **Grid**. Press F5 to run the program.

```
Private Sub Form_Load ()
   Grid1.Rows = 8 ' Set rows and columns.
   Grid1.Cols = 5
End Sub

Private Sub Form_Click ()
   Dim Msg As String
   Msg = "Date" & Chr(9)   ' Create a text string.
   Msg = Msg & Format(Now, "Short Date") & Chr(13)
   Msg = Msg & "Time" & Chr(9)
   Msg = Msg & Format(Now, "Short Time")
   Grid1.Clip = Msg  ' Paste string into grid.
End Sub
```

ClipControls Property Example

This example shows how the **ClipControls** property affects the repainting of a form. To try this example, paste the code into the Declarations section of a form, and then press F5. Notice that the color of the entire form changes each time you resize it or cover part of it with another form or application. End the program and set **ClipControls** to **False**, and then run the program again. Notice that only newly exposed parts of the form are repainted.

```
Private Sub Form_Paint ()
   ' Select a random color for the background.
   BackColor = &HFFFFFF * Rnd
End Sub
```

ColAlignment Property Example

This example fills a **Grid** control with the letter m and then sets the column alignment to right-aligned for every other column. To try this example, use the Components dialog box to add a **Grid** control to the toolbox (from the Project menu, choose OLE Components, and then select Microsoft Grid Control), and then draw a **Grid** on a new form. Press F5 to run the program.

```
Private Sub Form_Load ()
   ' Set Cols and Rows.
   Grid1.Cols = 5
   Grid1.Rows = 8
   ' Select all cells.
   Grid1.SelStartCol = 1
   Grid1.SelStartRow = 1
   Grid1.SelEndCol = Grid1.Cols - 1
   Grid1.SelEndRow = Grid1.Rows - 1
   ' Fill all cells with a letter.
   Grid1.FillStyle = 1  ' Turn on FillStyle.
   Grid1.Text = "m"
   ' Set ColAlignment for even numbered columns.
   For I = 2 To 4 Step 2
      Grid1.ColAlignment(I) = 1    ' Right align.
   Next I
   Grid1.FillStyle = 0  ' Turn off FillStyle.
End Sub
```

Columns Property Example

This example illustrates how the two different kinds of **ListBox** controls work when they contain the same data. To try this example, paste the code into the Declarations section of a form that contains two **ListBox** controls. Set the **Columns** property to 2 for List2, and then press F5 and click the form.

```
Private Sub Form_Load ()
   Dim I ' Declare variable.
   List1.Move 50, 50, 2000, 1750   ' Arrange list boxes.
   List2.Move 2500, 50, 3000, 1750
   For I = 0 To Screen.FontCount -1      ' Fill both boxes with
      List1.AddItem Screen.Fonts(I)      ' names of screen fonts.
      List2.AddItem Screen.Fonts(I)
   Next I
End Sub
```

ColWidth Property Example

This example enables the user to increase the column width of selected columns with the > key, and decrease the column width with the < key. To try this example, use the Components dialog box to add a **Grid** control to the toolbox (from the Project menu, choose Components, and then select Microsoft Grid Control), and then draw a **Grid** control.

```
Private Sub Form_Load ()
   ' Set columns and rows.
   Grid1.Cols = 6
   Grid1.Rows = 8
End Sub

Private Sub Grid1_KeyPress (KeyAscii As Integer)
   Select Case KeyAscii
      Case 62  ' > key.
         ' Get index for each selected column.
         For I = Grid1.SelStartCol To Grid1.SelEndCol
            ' Increase column width.
            Grid1.ColWidth(I) = Grid1.ColWidth(I) + 50
         Next I
      Case 60  ' < key.
         'Get index for each selected column.
         For I = Grid1.SelStartCol To Grid1.SelEndCol
            'Decrease column width.
            Grid1.ColWidth(I) = Grid1.ColWidth(I) - 50
         Next I
   End Select
End Sub
```

DragIcon Property Example

This example changes the **DragIcon** property setting each time you drag a **PictureBox** control. To try this example, paste the code into the Declarations section of a form that contains a **PictureBox** control. Set the **DragMode** property = 1, and then press F5 and click and drag the **PictureBox** control.

```
Private Sub Form_DragDrop (Source As Control, X As Single, Y As Single)
   Dim Pic  ' Declare variable.
   Source.Move X, Y  ' Set position of control.
   Pic = "ICONS\OFFICE\CRDFLE01.ICO"     ' Get name of icon file.
   If Source.DragIcon = False Then ' If no picture loaded,
      Source.DragIcon = LoadPicture(Pic) ' load picture.
   Else
      Source.DragIcon = LoadPicture()    ' Unload picture.
   End If
End Sub
```

DragMode Property Example

This example enables and disables the ability to drag a **CommandButton** control each time a form is clicked. To try this example, paste the code into the Declarations section of a form that contains a **CommandButton**, and then press F5 and click the form.

```
Private Sub Form_Click ()
   ' Check DragMode.
   If Command1.DragMode = vbManual Then
      ' Turn it on.
      Command1.DragMode = vbAutomatic
   Else
      ' Or turn it off.
      Command1.DragMode = vbManual
   End If
End Sub
```

DrawMode Property Example

This example enables drawing on a form by dragging the mouse pointer. Each mouse click sets a different value for the **DrawMode** property. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Load
   DrawWidth = 10 ' Set DrawWidth.
End Sub
Private Sub Form_Click ()
   Static M As Integer  ' Current DrawMode setting.
   ForeColor = QBColor(Int(Rnd * 15))    ' Choose a color.
   M = ((M + 1) Mod 16) + 1  ' Keep DrawMode 16 or less.
   DrawMode = M   ' Set DrawMode.
End Sub
Private Sub Form_MouseMove (Button As Integer, Shift As Integer, X As
Single, Y As Single)
   If Button Then ' While button is pressed,
      PSet (X, Y) ' draw a big point.
   End If
End Sub
```

DrawStyle Property Example

This example draws seven lines across a form, with each line displaying a different **DrawStyle** property. (If you set **AutoRedraw** = **True**, the form accumulates a new set of lines each time you resize it and then click it.) To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim I ' Declare variable.
   ScaleHeight = 8   ' Divide height by 8.
   For I = 0 To 6
      DrawStyle = I  ' Change style.
      Line (0, I + 1) - (ScaleWidth, I + 1)    ' Draw new line.
   Next I
End Sub
```

Drive Property Example

This example displays a list of files for the current drive and directory. To try this example, paste the code into the Declarations section of a form that contains a **DriveListBox** control, a **DirListBox** control, and a **FileListBox** control, and then press F5. Use the mouse to change the drive or directory.

```
Private Sub Drive1_Change ()
   Dir1.Path = Drive1.Drive  ' When drive changes, set directory path.
End Sub
Private Sub Dir1_Change ()
   File1.Path = Dir1.Path  ' When directory changes, set file path.
End Sub
```

FileName Property Example

This example displays a message in a **Label** control when a filename in a **FileListBox** control is double-clicked. To try this example, paste the code into the Declarations section of a form that contains a **Label** control, a **DirListBox** control, and a **FileListBox** control, and then press F5 and double-click any filename in the **FileListBox** control.

```
Private Sub Dir1_Change ()
   File1.Path = Dir1.Path  ' Set File1 path.
End Sub
Private Sub File1_PathChange ()
   Dir1.Path = File1.Path  ' Set Dir1 path.
End Sub
Private Sub File1_DblClick ()
   ' Display the selected filename when double-clicked.
   Label1.Caption = "Your selection: " +_ File1.FileName
End Sub
```

FillColor Property Example

This example constructs a circle on your form with random **FillColor** and **FillStyle** property settings as you click the mouse. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_MouseDown (Button As Integer, Shift As Integer, X As
Single, Y As Single)
   FillColor = QBColor(Int(Rnd * 15))     ' Choose random FillColor.
   FillStyle = Int(Rnd * 8)  ' Choose random FillStyle.
   Circle (X, Y), 250    ' Draw a circle.
End Sub
```

FillStyle Property Example

This example displays a circle on a form with random **FillColor** and **FillStyle** property settings as you click the mouse. To try this example, paste the code into the Declarations section, and then press F5 to run the program.

```
Private Sub Form_MouseDown (Button As Integer, Shift As Integer, X As
Single, Y As Single)
   FillColor = QBColor(Rnd * 15)   ' Choose random FillColor.
   FillStyle = Int(Rnd * 8)  ' Choose random FillStyle.
   Circle (X, Y), 250   ' Draw a circle.
End Sub
```

FixedAlignment Property Example

This example creates a spreadsheet with centered column headings and row numbers. To try this example, use the Components dialog box to add a **Grid** control to the toolbox (from the Project menu, choose Components, and then select Microsoft Grid Control), and then draw a **Grid** control. Press F5 to run the program.

```
Private Sub Form_Load ()
   Grid1.Cols = 5 ' Set rows and columns.
   Grid1.Rows = 7
   Grid1.Row = 0
   Grid1.FixedAlignment(0) = 2     ' Set alignment of row numbers.
   For I = 1 To Grid1.Cols - 1     ' Set column letters and alignment.
      Grid1.FixedAlignment(I) = 2
      Grid1.Col = I
      Grid1.Text = Chr(64 + I)
   Next I
   Grid1.Col = 0
   For I = 1 To Grid1.Rows - 1     ' Set row numbers.
      Grid1.Row = I
      Grid1.Text = I
   Next I
End Sub
```

FixedCols, FixedRows Properties Example

This example illustrates a **Grid** control with two fixed rows and two fixed columns. To try this example, create a new project, use the Components dialog box to add a **Grid** control to the toolbox (from the Project menu, choose Components, and then select Microsoft Grid Control), and then draw a **Grid**. To run the program, press F5.

```
Private Sub Form_Load ()
   Grid1.Rows = 8 ' Set columns and rows.
   Grid1.Cols = 6
   Grid1.FixedRows = 2  ' Set fixed columns and rows.
   Grid1.FixedCols = 2
End Sub
```

FontCount Property Example

This example prints a list of the printer fonts in a **ListBox** control. To try this example, paste the code into the Declarations section of a form that has a **ListBox** control named List1, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim I ' Declare variable.
   For I = 0 To Printer.FontCount -1      ' Determine number of fonts.
      List1.AddItem Printer.Fonts (I)     ' Put each font into list box.
   Next I
End Sub
```

Fonts Property Example

This example prints a list of the printer fonts in a **ListBox** control. To try this example, paste the code into the Declarations section of a form that has a **ListBox** control named List1. Press F5 to run the program, and then click the form.

```
Private Sub Form_Click ()
   Dim I ' Declare variable.
   For I = 0 To Printer.FontCount -1     ' Determine number of fonts.
      List1.AddItem Printer.Fonts (I)    ' Put each font into list box.
   Next I
End Sub
```

GridLines Property Example

This example turns the **GridLines** property on and off as you click the form. To try this example, create a new project, use the Components dialog box to add a **Grid** control to the toolbox (from the Project menu, choose Components, and then select Microsoft Grid Control), and then draw a **Grid** control. Copy the code into the Declarations section, and then press F5 to run the program.

```
Private Sub Form_Load ()
   Grid1.Cols = 6 ' Set columns and rows.
   Grid1.Rows = 7
End Sub

Private Sub Form_Click ()  ' Toggle gridlines on and off.
   Grid1.GridLines = Not Grid1.GridLines
End Sub
```

hDC Property Example

This example draws a triangle and then uses a Microsoft Windows function to fill it with color. To try this example, create a new module using the Add Module command on the Project menu. Paste the **Declare** statement into the Declarations section of the new module, being sure that the statement is on one line with no break or wordwrap. Then paste the **Sub** procedure into the Declarations section of a form. Press F5 and click the form.

```
' Declaration of a Windows routine. This statement is for the module.
Declare Sub FloodFill Lib "GDI" (ByVal hDC As Integer, ByVal X As Integer,
ByVal Y As Integer, ByVal Color As Long)
' The following code is for the form.
Private Sub Form_Click ()
   ScaleMode = vbPixels     ' Windows draws in pixels.
   ForeColor = vbBlack      ' Set draw line to black.
   Line (100, 50)-(300, 50) ' Draw a triangle.
   Line -(200, 200)
   Line -(100, 50)
   FillStyle = vbFSSolid    ' Set FillStyle to solid.
   FillColor = RGB(128, 128, 255)     ' Set FillColor.
   ' Call Windows API to fill.
   FloodFill hDC, 200, 100, ForeColor
End Sub
```

HideSelection Property Example

This example enables you to select text in each form and switch the focus between forms by clicking each form's title bar. The selection remains visible even when the form isn't active. To try the example, create two forms and draw a **TextBox** control on each. Set the **MultiLine** property to **True** for both **TextBox** controls, and set the **HideSelection** property to **False** for one of the **TextBox** controls. Paste the code into the Declarations section of both form modules, and then press F5.

```
Private Sub Form_Load ()
   Open "README.TXT" For Input As 1      ' Load file into text box.
   Text1.Text = Input$(LOF(1), 1)
   Close 1
   Form2.Visible = True ' Load Form2, if not already loaded.
   ' Position forms side by side.
   Form1.Move 0, 1050, Screen.Width / 2, Screen.Height
   Form2.Move Screen.Width / 2, 1050, Screen.Width / 2, Screen.Height
   ' Enlarge text box to fill form.
   Text1.Move 0, 0, ScaleWidth, ScaleHeight
End Sub
```

HighLight Property Example

This example turns highlighting on and off as you click the form. To try this example, create a new project, use the Components dialog box to add a **Grid** control to the toolbox (from the Project menu, choose Components, and then select Microsoft Grid Control), and then draw a **Grid** control. Copy the code into the Declarations section, and then press F5 to run the program.

```
Private Sub Form_Load ()
   Grid1.Cols = 6 ' Set columns and rows.
   Grid1.Rows = 7
End Sub

Private Sub Form_Click ()  ' Toggle highlighting on and off.
   Grid1.HighLight = Not Grid1.HighLight
End Sub
```

hWnd Property Example

This example forces a form to always remain on top. To try this example, create a form (not an MDI child form), and then create a menu for the form called Main. Insert a submenu in it called Always On Top, and set its Name to mnuTopmost. Create a new module using the Module command on the Insert menu. Paste the **Declare** statement into the Declarations section of the new module, being sure that the statement is on one line with no break or wordwrap. Then paste the **Sub** procedure into the Declarations section of the form and press F5.

```
' Declaration of a Windows routine.
' This statement should be placed in the module.
Declare Function SetWindowPos Lib "user32" Alias_ "SetWindowPos" (ByVal
hwnd As Long, ByVal_ hWndInsertAfter As Long, ByVal x As Long, ByVal y As_
Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags_ As Long) As Long

' Set some constant values (from WINAPI.TXT).
Const conHwndTopmost = -1
Const conHwndNoTopmost = -2
Const conSwpNoActivate = &H10
Const conSwpShowWindow = &H40

Private Sub mnuTopmost_Click ()
   ' Add or remove the check mark from the menu.
   mnuTopmost.Checked = Not mnuTopmost.Checked
   If mnuTopmost.Checked Then
      ' Turn on the TopMost attribute.
      SetWindowPos hWnd, conHwndTopmost, 0, 0, 0, 0,_ conSwpNoActivate Or
conSwpShowWindow
   Else
      ' Turn off the TopMost attribute.
      SetWindowPos hWnd, conHwndNoTopmost, 0, 0, 0,_ 0, conSwpNoActivate
Or conSwpShowWindow
   End If
End Sub
```

This example automatically drops down the list portion of a **ComboBox** control whenever the **ComboBox** receives the focus. To try this example, create a new form containing a **ComboBox** control and an **OptionButton** control (used only to receive the focus). Create a new module using the Module command on the Insert menu. Paste the **Declare** statement into the Declarations section of the new module, being sure that the statement is on one line with no break or wordwrap. Then paste the **Sub** procedure into the Declarations section of the form, and press F5. Use the TAB key to move the focus to and from the **ComboBox**.

```
Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd
As Long, ByVal wMsg As Long, ByVal wParam As Long, lParam As Long) As Long

Private Sub Combo1_GotFocus ()
   Const CB_SHOWDROPDOWN = &H14F
   Dim Tmp
   Tmp = SendMessage(Combo1.hWnd, CB_SHOWDROPDOWN, 1, ByVal 0&)
End Sub
```

Image Property Example

This example draws a circle in the first **PictureBox** control each time you click it. When you click the second **PictureBox**, the graphic from the first **PictureBox** is copied into it. To try this example, paste the code into the Declarations section of a form that has two large, equal-sized **PictureBox** controls. Press F5 to run the program, and then click the **PictureBox** controls.

```
Private Sub Form_Load ()
   ' Set AutoRedraw to True.
   Picture1.AutoReDraw = True
End Sub

Private Sub Picture1_Click ()
   ' Declare variables.
   Dim PW, PH
   ' Set FillStyle to Solid.
   Picture1.FillStyle = vbFSSolid
   ' Choose random color.
   Picture1.FillColor = QBColor(Int(Rnd * 15))
   PW = Picture1.ScaleWidth  ' Set ScaleWidth.
   PH = Picture1.ScaleHeight ' Set ScaleHeight.
   ' Draw a circle in random location.
   Picture1.Circle (Int(Rnd * PW), Int(Rnd * PH)), 250
End Sub

Private Sub Picture2_Click ()
   ' Copy Image to Picture2.
   Picture2.Picture = Picture1.Image
End Sub
```

ItemData Property Example

This example fills a **ListBox** control with employee names and fills the **ItemData** property array with employee numbers using the **NewIndex** property to keep the numbers synchronized with the sorted list. A **Label** control displays the name and number of an item when the user makes a selection. To try this example, paste the code into the Declarations section of a form that contains a **ListBox** and a **Label**. Set the **Sorted** property for the **ListBox** to **True**, and then press F5 and click the **ListBox**.

```
Private Sub Form_Load ()
   ' Fill List1 and ItemData array with
   ' corresponding items in sorted order.
   List1.AddItem "Judy Phelps"
   List1.ItemData(List1.NewIndex) = 42310
   List1.AddItem "Chien Lieu"
   List1.ItemData(List1.NewIndex) = 52855
   List1.AddItem "Mauro Sorrento"
   List1.ItemData(List1.NewIndex) = 64932
   List1.AddItem "Cynthia Bennet"
   List1.ItemData(List1.NewIndex) = 39227
End Sub

Private Sub List1_Click ()
   ' Append the employee number and the employee name.
   Msg = List1.ItemData(List1.ListIndex) & " "
   Msg = Msg & List1.List(List1.ListIndex)
   Label1.Caption = Msg
End Sub
```

KeyPreview Property Example

This example creates a form keyboard handler in the KeyDown event. Each of the first four function keys displays a different message. To try this example, paste the code into the Declarations section of a form, and then press F5. Once the program is running, press any one of the first four (F1 – F4) function keys.

```
Private Sub Form_Load ()
   KeyPreview = True
End Sub

Private Sub Form_KeyDown (KeyCode As Integer, Shift As Integer)
   Select Case KeyCode
      Case vbKeyF1: MsgBox "F1 is your friend."
      Case vbKeyF2: MsgBox "F2 could copy text."
      Case vbKeyF3: MsgBox "F3 could paste text."
      Case vbKeyF4: MsgBox "F4 could format text."
   End Select
End Sub
```

LargeChange, SmallChange Properties Example

This example uses a scroll bar to move a **PictureBox** control across the form. To try this example, paste the code into the Declarations section of a form that contains a small **PictureBox** control and an **HScrollBar** control, and then press F5 and click the scroll bar.

```
Private Sub Form_Load ()
   HScroll1.Max = 100   ' Set maximum value.
   HScroll1.LargeChange = 20 ' Cross in 5 clicks.
   HScroll1.SmallChange = 5  ' Cross in 20 clicks.
   Picture1.Left = 0 ' Start picture at left.
   Picture1.BackColor = QBColor(3) ' Set color of picture box.
End Sub
Private Sub HScroll1_Change ()
   ' Move picture according to scroll bar.
   Picture1.Left = (HScroll1.Value / 100) * ScaleWidth
End Sub
```

LeftCol Property Example

This example changes the **LeftCol** property with each click of the form. To try this example, use the Components dialog box to add a **Grid** control to the toolbox (from the Project menu, choose Components, and then select Microsoft Grid Control), and then draw a **Grid** control. Copy the code into the Declarations section, and then press F5 to run the program.

```
Private Sub Form_Load ()
   Grid1.Rows = 8 ' Set columns and rows.
   Grid1.Cols = 9
   Grid1.Row = 0
   For I = 1 To Grid1.Cols - 1     ' Set numbers in column heads.
      Grid1.Col = I
      Grid1.Text = I
   Next I
End Sub

Private Sub Form_Click ()
   On Error GoTo LeftColError
   ' Scroll the grid by one column.
   Grid1.LeftCol = Grid1.LeftCol + 1
   On Error GoTo 0
   Exit Sub

LeftColError:
   Grid1.LeftCol = 1 ' Display the first column.
   Resume Next
End Sub
```

LinkItem, LinkMode, LinkTopic Properties Example

In the example, each mouse click causes a cell in a Microsoft Excel worksheet to update the contents of a Visual Basic **TextBox** control. To try this example, start Microsoft Excel, open a new worksheet named Sheet1, and put some data in the first column. In Visual Basic, create a form with a **TextBox** control. Paste the code into the Declarations section, and then press F5 to run the program.

```
Private Sub Form_Click ()
   Dim CurRow As String
   Static Row  ' Worksheet row number.
   Row = Row + 1  ' Increment Row.
   If Row = 1 Then   ' First time only.
      ' Make sure the link isn't active.
      Text1.LinkMode = 0
      ' Set the application name and topic name.
      Text1.LinkTopic = "Excel|Sheet1"
      Text1.LinkItem = "R1C1"      ' Set LinkItem.
      Text1.LinkMode = 1   ' Set LinkMode to Automatic.
   Else
      ' Update the row in the data item.
      CurRow = "R" & Row & "C1"
      Text1.LinkItem = CurRow      ' Set LinkItem.
   End If
End Sub
```

GridLineWidth Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproGridLineWidthC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproGridLineWidthX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproGridLineWidthA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproGridLineWidthS"}

Returns or sets the width in pixels of the gridlines for a **Grid** control.

Syntax

object.**GridLineWidth** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer specifying the gridline width. The minimum setting is 1 (default); the maximum setting is 10.

LinkTopic Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLinkTopicC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLinkTopicX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLinkTopicA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLinkTopicS"}

For a destination control — returns or sets the source application and the topic (the fundamental data grouping used in that application). Use **LinkTopic** with the **LinkItem** property to specify the complete data link.

For a source form — returns or sets the topic that the source form responds to in a DDE conversation.

Syntax

object.**LinkTopic** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A string expression specifying a DDE syntax element.

Remarks

The **LinkTopic** property consists of a string that supplies part of the information necessary to set up either a destination link or source link. The string you use depends on whether you're working with a destination control or a source form. Each string corresponds to one or more elements of standard DDE syntax, which include *application*, *topic*, and *item*.

Note While the standard definition for a DDE link includes the *application*, *topic*, and *item* elements, the actual syntax used within applications for a destination link to a source application may vary slightly. For example, within Microsoft Excel, you use the syntax:

*application***|***topic***!***item*

Within Microsoft Word for Windows, you use:

application topic item

(Don't use the pipe character [**|**] or exclamation mark [**!**].)

Within a Visual Basic application, you use:

*application***|***topic*

The exclamation mark for *topic* is implicit.

Destination Control To set **LinkTopic** for a destination control, use a string with the syntax *application***|***topic* as follows:

- *application* is the name of the application from which data is requested, usually the executable filename without an extension — for example, Excel (for Microsoft Excel).
- The pipe character (**|**, or character code 124) separates the application from the topic.
- *topic* is the fundamental data grouping used in the source application — for example, a worksheet in Microsoft Excel.

In addition, for a destination control only, you must set the related **LinkItem** property to specify the *item* element for the link. A cell reference, such as R1C1, corresponds to an item in a Microsoft Excel worksheet.

Source Form To set **LinkTopic** for a source form, set *value* to an appropriate identifier for the form. A destination application uses this string as the *topic* argument when establishing a DDE link with the

form. Although this string is all you need to set **LinkTopic** within Visual Basic for a source form, the destination application also needs to specify:

- The *application* element that the destination application uses, which is either the Visual Basic project filename without the .vbp extension (if you're running your application in the Visual Basic development environment) or the Visual Basic application filename without the .exe extension (if you're running your application as a stand-alone executable file). The **EXEName** property of the **App** object provides this string in your Visual Basic code unless the filename was changed by the user. (**EXEName** always returns the actual filename of the application on disk; DDE always uses the original name that was specified in the Project Properties dialog box.)
- The *item* element that the destination application uses, which corresponds to the **Name** property setting for the **Label**, **PictureBox**, or **TextBox** control on the source form.

The following syntax is an example of a valid reference from Microsoft Excel to a Visual Basic application acting as a source:

=*VizBasicApplication*|*FormN*!*TextBox1*

You could enter this reference for a destination cell in the Microsoft Excel formula bar.

To activate the data link set with **LinkTopic**, set the **LinkMode** property to the appropriate nonzero value to specify the type of link you want. As a general rule, set **LinkMode** after you set **LinkTopic**. For a destination control, changing **LinkTopic** breaks an existing link and terminates the DDE conversation. For a source form, changing **LinkTopic** breaks all destination links that are using that topic. For these reasons, always set the **LinkMode** property to 0 before changing **LinkTopic**. After changing **LinkTopic** for a destination control, you must set **LinkMode** to 1 (Automatic), 2 (Manual), or 3 (Notify) to establish a conversation with the new topic.

Note Setting a permanent data link at design time with the Paste Link command on the Edit menu also sets the **LinkMode**, **LinkTopic**, and **LinkItem** properties. This creates a link that is saved with the form. Each time the form is loaded, Visual Basic attempts to reestablish the conversation.

Max, Min Properties (Scroll Bar)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMaxScrollC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMaxScrollX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMaxScrollA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMaxScrollS"}

- **Max** — returns or sets a scroll bar position's maximum **Value** property setting when the scroll box is in its bottom or rightmost position. For the ProgressBar control, it returns or sets its maximum value.
- **Min** — returns or sets a scroll bar position's minimum **Value** property setting when the scroll box is in its top or leftmost position. For the ProgressBar control, it returns or sets its minimum value.

Syntax

object.**Max** [**=** *value*]
object.**Min** [**=** *value*]

The **Max** and **Min** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A numeric expression specifying the maximum or minimum **Value** property setting, as described in Settings.

Settings

For each property, you can specify an integer between -32,768 and 32,767, inclusive. The default settings are:

- **Max** — 32,767.
- **Min** — 0.

Remarks

The Microsoft Windows operating environment automatically sets ranges for scroll bars proportional to the contents of forms, **ComboBox** controls, and **ListBox** controls. For a scroll bar (**HScrollBar** or **VScrollBar)** control, however, you must specify these ranges. Use **Max** and **Min** to set a range appropriate to how the scroll bar control is used — for example, as an input device or as an indicator of speed or quantity.

Typically, you set **Max** and **Min** at design time. You can also set them in code at run time if the scrolling range must change dynamically — for example, when adding records to a database that can be scrolled through. You set the maximum and minimum scrolling increments for a scroll bar control with the **LargeChange** and **SmallChange** properties.

Note If **Max** is set to less than **Min**, the maximum value is set at the leftmost or topmost position of a horizontal or vertical scroll bar, respectively. The **Max** property of a **ProgressBar** control must always be greater than its **Min** property, and its **Min** property must always be greater than or equal to 0.

The **Max** and **Min** properties define the range of the control. The **ProgressBar** control's **Min** property is 0 and its **Max** property is 100 by default, representing the percentage duration of the operation.

MaxLength Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMaxLengthC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMaxLengthX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMaxLengthA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMaxLengthS"}

Returns or sets a value indicating whether there is a maximum number of characters that can be entered in the **TextBox** control and, if so, specifies the maximum number of characters that can be entered.

Note In DBCS systems, each character can take up to two bytes instead of only one, which limits the number of characters you can enter.

Syntax

object.**MaxLength** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer specifying the maximum number of characters a user can enter in a **TextBox** control. The default for the **MaxLength** property is 0, indicating no maximum other than that created by memory constraints on the user's system for single-line **TextBox** controls and a maximum of approximately 32K for multiple-line **TextBox** controls. Any number greater than 0 indicates the maximum number of characters.

Remarks

Use the **MaxLength** property to limit the number of characters a user can enter in a **TextBox**.

If text that exceeds the **MaxLength** property setting is assigned to a **TextBox** from code, no error occurs; however, only the maximum number of characters is assigned to the **Text** property, and extra characters are truncated. Changing this property doesn't affect the current contents of a **TextBox** but will affect any subsequent changes to the contents.

MDIChild Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMDIChildC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMDIChildX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMDIChildA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMDIChildS"}

Returns or sets a value indicating whether a form is displayed as an MDI child form inside an MDI form. Read only at run time.

Syntax

object.**MDIChild**

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

Settings

The **MDIChild** property settings are:

Setting	Description
True	The form is an MDI child form and is displayed inside the parent MDI form.
False	(Default) The form isn't an MDI child form.

Remarks

Use this property when creating a multiple-document interface (MDI) application. At run time, forms with this property set to **True** are displayed inside an MDI form. An MDI child form can be maximized, minimized, and moved, all inside the parent MDI form.

When working with MDI child forms, keep the following in mind:

- At run time, when an MDI child form is maximized, its caption is combined with that of the parent MDI form.
- At design time, an MDI child form is displayed like any other form because the form is displayed inside the parent form only at run time. An MDI child form's icon in the Project window is different from icons for other kinds of forms.
- MDI child forms can't be modal.
- The initial size and placement of MDI child forms are controlled by the Microsoft Windows operating environment unless you specifically set them in the Load event procedure.
- If an MDI child form is referenced before the parent is loaded, the parent MDI form is automatically loaded. However, if the parent MDI form is referenced before loading an MDI child form, the child form isn't loaded.

Note All MDI child forms have sizable borders, a Control-menu box, and Minimize and Maximize buttons, regardless of the settings of the **BorderStyle**, **ControlBox**, **MinButton**, and **MaxButton** properties.

Any reference to an **MDIForm** object, including reading or setting properties, causes the form to load and become visible.

MouseIcon Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMouseIconC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMouseIconX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMouseIconA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMouseIconS"}

Returns or sets a custom mouse icon.

Syntax

object.**MouseIcon = LoadPicture(***pathname***)**
object.**MouseIcon [=** *picture***]**

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
pathname	A string expression specifying the path and filename of the file containing the custom icon.
picture	The **Picture** property of a **Form** object, **PictureBox** control, or **Image** control.

Remarks

The **MouseIcon** property provides a custom icon that is used when the **MousePointer** property is set to 99.

Although Visual Basic does not create or support color cursor (.cur) files (such as those that ship with Windows NT), you can use the **MouseIcon** property to load either cursor or icon files. Color cursor files such as those shipped with Windows NT 3.51, are displayed in black and white. To display a color cursor, use a color icon file (.ico). The **MouseIcon** property provides your program with easy access to custom cursors of any size, with any desired hot spot location. Visual Basic does not load animated cursor (.ani) files, even though 32-bit versions of Windows support these cursors.

MousePointer Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMousePointerC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMousePointerX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMousePointerA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMousePointerS"}

Returns or sets a value indicating the type of mouse pointer displayed when the mouse is over a particular part of an object at run time.

Syntax

object.**MousePointer** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer specifying the type of mouse pointer displayed, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
vbDefault	0	(Default) Shape determined by the object.
VbArrow	1	Arrow.
VbCrosshair	2	Cross (crosshair pointer).
VbIbeam	3	I beam.
VbIconPointer	4	Icon (small square within a square).
VbSizePointer	5	Size (four-pointed arrow pointing north, south, east, and west).
VbSizeNESW	6	Size NE SW (double arrow pointing northeast and southwest).
VbSizeNS	7	Size N S (double arrow pointing north and south).
VbSizeNWSE	8	Size NW SE (double arrow pointing northwest and southeast).
VbSizeWE	9	Size W E (double arrow pointing west and east).
VbUpArrow	10	Up Arrow.
VbHourglass	11	Hourglass (wait).
VbNoDrop	12	No Drop.
VbArrowHourglass	13	Arrow and hourglass. (Only available in 32-bit Visual Basic.)
vbArrowQuestion	14	Arrow and question mark. (Only available in 32-bit Visual Basic.)
vbSizeAll	15	Size all. (Only available in 32-bit Visual Basic.)
vbCustom	99	Custom icon specified by the **MouseIcon** property.

Remarks

You can use this property when you want to indicate changes in functionality as the mouse pointer passes over controls on a form or dialog box. The Hourglass setting (11) is useful for indicating that the user should wait for a process or operation to finish.

Note If your application calls DoEvents, the **MousePointer** property may temporarily change when over a custom control.

MultiLine Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMultiLineC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMultiLineX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMultiLineA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMultiLineS"}

Returns or sets a value indicating whether a **TextBox** control can accept and display multiple lines of text. Read only at run time.

Syntax

object.**MultiLine**

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

Settings

The **MultiLine** property settings are:

Setting	Description
True	Allows multiple lines of text.
False	(Default) Ignores carriage returns and restricts data to a single line.

Remarks

A multiple-line **TextBox** control wraps text as the user types text extending beyond the text box.

You can also add scroll bars to larger **TextBox** controls using the **ScrollBars** property. If no horizontal scroll bar is specified, the text in a multiple-line **TextBox** automatically wraps.

Note On a form with no default button, pressing ENTER in a multiple-line **TextBox** control moves the focus to the next line. If a default button exists, you must press CTRL+ENTER to move to the next line.

MultiSelect Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMultiSelectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMultiSelectX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMultiSelectA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMultiSelectS"}

Returns or sets a value indicating whether a user can make multiple selections in a **FileListBox** or **ListBox** control and how the multiple selections can be made. Read only at run time.

Syntax

object.**MultiSelect**

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

Settings

The **MultiSelect** property settings are:

Setting	Description
0	(Default) Multiple selection isn't allowed.
1	Simple multiple selection. A mouse click or pressing the SPACEBAR selects or deselects an item in the list. (Arrow keys move the focus.)
2	Extended multiple selection. Pressing SHIFT and clicking the mouse or pressing SHIFT and one of the arrow keys (UP ARROW, DOWN ARROW, LEFT ARROW, and RIGHT ARROW) extends the selection from the previously selected item to the current item. Pressing CTRL and clicking the mouse selects or deselects an item in the list.

NewIndex Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproNewIndexC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproNewIndexX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproNewIndexA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproNewIndexS"}

Returns the index of the item most recently added to a **ComboBox** or **ListBox** control. Read only at run time.

Syntax

object.**NewIndex**

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

Remarks

You can use this property with sorted lists when you need a list of values that correspond to each item in the **ItemData** property array. As you add an item in a sorted list, Visual Basic inserts the item in the list in alphabetic order. This property tells you where the item was inserted so that you can insert a corresponding value in the **ItemData** property at the same index.

The **NewIndex** property returns -1 if there are no items in the list or if an item has been deleted since the last item was added.

Orientation Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproOrientationC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOrientationX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOrientationA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOrientationS"}

Returns or sets a value indicating whether documents are printed in portrait or landscape mode. Not available at design time.

Syntax

object.**Orientation** [**=** *value*]

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

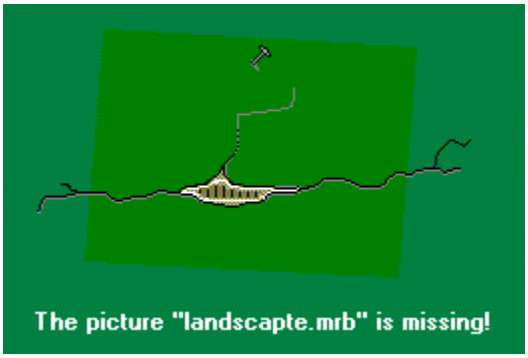
Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A value or constant that determines the page orientation, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
vbPRORPortrait	1	Documents are printed with the top at the narrow side of the paper.

vbPRORLandscape	2	Documents are printed with the top at the wide side of the paper.



Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

Note The effect of the properties of the **Printer** object depends on the driver supplied by the printer manufacturer. Some property settings may have no effect, or several different property settings may all have the same effect. Settings outside the accepted range may or may not produce an error. For more information, see the manufacturer's documentation for the specific driver.

Page Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPageC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPageX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPageA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPageS"}

Returns the current page number.

Syntax

object.**Page**

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

Remarks

Visual Basic keeps a count of pages that have been printed since your application started or since the last time the **EndDoc** statement was used on the **Printer** object. This count starts at one and increases by one if:

- You use the **NewPage** method.
- You use the **Print** method and the text you want to print doesn't fit on the current page.

Note Graphics methods output that doesn't fit on the page doesn't generate a new page. The output is clipped to fit the page's printable area.

PaperBin Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPaperBinC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPaperBinX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPaperBinA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPaperBinS"}

Returns or sets a value indicating the default paper bin on the printer from which paper is fed when printing. Not available at design time.

Syntax

object.**PaperBin** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A value or constant specifying the default paper bin, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
vbPRBNUpper	1	Use paper from the upper bin.
VbPRBNLower	2	Use paper from the lower bin.
VbPRBNMiddle	3	Use paper from the middle bin.
VbPRBNManual	4	Wait for manual insertion of each sheet of paper.
VbPRBNEnvelope	5	Use envelopes from the envelope feeder.
VbPRBNEnvManual	6	Use envelopes from the envelope feeder, but wait for manual insertion.
VbPRBNAuto	7	(Default) Use paper from the current default bin.
VbPRBNTractor	8	Use paper fed from the tractor feeder.
VbPRBNSmallFmt	9	Use paper from the small paper feeder.
VbPRBNLargeFmt	10	Use paper from the large paper bin.
VbPRBNLargeCapacity	11	Use paper from the large capacity feeder.
VbPRBNCassette	14	Use paper from the attached cassette cartridge.

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

Not all of the bin options are available on every printer. Check the printer documentation for more specific descriptions of these options.

Note The effect of the properties of the **Printer** object depends on the driver supplied by the printer manufacturer. Some property settings may have no effect, or several different property settings may all have the same effect. Settings outside the accepted range may or may not produce an error. For

more information, see the manufacturer's documentation for the specific driver.

PaperSize Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPaperSizeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPaperSizeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPaperSizeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPaperSizeS"}

Returns or sets a value indicating the paper size for the current printer. Not available at design time.

Syntax

object.**PaperSize** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A value or constant specifying the paper size, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
vbPRPSLetter	1	Letter, 8 1/2 x 11 in.
vbPRPSLetterSmall	2	Letter Small, 8 1/2 x 11 in.
vbPRPSTabloid	3	Tabloid, 11 x 17 in.
vbPRPSLedger	4	Ledger, 17 x 11 in.
vbPRPSLegal	5	Legal, 8 1/2 x 14 in.
vbPRPSStatement	6	Statement, 5 1/2 x 8 1/2 in.
vbPRPSExecutive	7	Executive, 7 1/2 x 10 1/2 in.
vbPRPSA3	8	A3, 297 x 420 mm
vbPRPSA4	9	A4, 210 x 297 mm
vbPRPSA4Small	10	A4 Small, 210 x 297 mm
vbPRPSA5	11	A5, 148 x 210 mm
vbPRPSB4	12	B4, 250 x 354 mm
vbPRPSB5	13	B5, 182 x 257 mm
vbPRPSFolio	14	Folio, 8 1/2 x 13 in.
vbPRPSQuarto	15	Quarto, 215 x 275 mm
vbPRPS10x14	16	10 x 14 in.
vbPRPS11x17	17	11 x 17 in.
vbPRPSNote	18	Note, 8 1/2 x 11 in.
vbPRPSEnv9	19	Envelope #9, 3 7/8 x 8 7/8 in.
vbPRPSEnv10	20	Envelope #10, 4 1/8 x 9 1/2 in.
vbPRPSEnv11	21	Envelope #11, 4 1/2 x 10 3/8 in.
vbPRPSEnv12	22	Envelope #12, 4 1/2 x 11 in.
vbPRPSEnv14	23	Envelope #14, 5 x 11 1/2 in.
vbPRPSCSheet	24	C size sheet

vbPRPSDSheet	25	D size sheet
vbPRPSESheet	26	E size sheet
vbPRPSEnvDL	27	Envelope DL, 110 x 220 mm
vbPRPSEnvC3	29	Envelope C3, 324 x 458 mm
vbPRPSEnvC4	30	Envelope C4, 229 x 324 mm
vbPRPSEnvC5	28	Envelope C5, 162 x 229 mm
vbPRPSEnvC6	31	Envelope C6, 114 x 162 mm
vbPRPSEnvC65	32	Envelope C65, 114 x 229 mm
vbPRPSEnvB4	33	Envelope B4, 250 x 353 mm
vbPRPSEnvB5	34	Envelope B5, 176 x 250 mm
vbPRPSEnvB6	35	Envelope B6, 176 x 125 mm
vbPRPSEnvItaly	36	Envelope, 110 x 230 mm
vbPRPSEnvMonarch	37	Envelope Monarch, 3 7/8 x 7 1/2 in.
vbPRPSEnvPersonal	38	Envelope, 3 5/8 x 6 1/2 in.
vbPRPSFanfoldUS	39	U.S. Standard Fanfold, 14 7/8 x 11 in.
vbPRPSFanfoldStdGerman	40	German Standard Fanfold, 8 1/2 x 12 in.
vbPRPSFanfoldLglGerman	41	German Legal Fanfold, 8 1/2 x 13 in.
vbPRPSUser	256	User-defined

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

Setting a printer's **Height** or **Width** property automatically sets **PaperSize** to **vbPRPSUser**.

Note The effect of the properties of the **Printer** object depends on the driver supplied by the printer manufacturer. Some property settings may have no effect, or several different property settings may all have the same effect. Settings outside the accepted range may or may not produce an error. For more information, see the manufacturer's documentation for the specific driver.

PasswordChar Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPasswordCharC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPasswordCharX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPasswordCharA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPasswordCharS"}

Returns or sets a value indicating whether the characters typed by a user or placeholder characters are displayed in a **TextBox** control; returns or sets the character used as a placeholder.

Syntax

object.**PasswordChar** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A string expression specifying the placeholder character.

Remarks

Use this property to create a password field in a dialog box. Although you can use any character, most Windows-based applications use the asterisk (*) (**Chr**(42)).

This property doesn't affect the **Text** property; **Text** contains exactly what the user types or what was set from code. Set **PasswordChar** to a zero-length string (""), which is the default, to display the actual text.

You can assign any string to this property, but only the first character is significant; all others are ignored.

Note If the **MultiLine** Property is set to **True**, setting the **PasswordChar** property will have no effect.

Pattern Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPatternC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPatternX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPatternA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPatternS"}

Returns or sets a value indicating the filenames displayed in a **FileListBox** control at run time.

Syntax

object.**Pattern** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A string expression indicating a file specification, such as "*.*" or "*.FRM". The default is "*.*", which returns a list of all files. In addition to using wildcard characters, you can also use multiple patterns separated by semicolons (;). For example, "*.exe; *.bat" would return a list of all executable files and all MS-DOS batch files.

Remarks

The **Pattern** property plays a key role in designing an application's file-browsing and manipulation capabilities. Use **Pattern** in combination with other file-control properties to provide the user with ways to explore files or groups of similar files. For example, in an application dedicated to launching other programs, you could designate that only .exe files be displayed in the file list box (*.exe). Other key file-control properties include **Drive**, **FileName**, and **Path**.

Changing the value of the **Pattern** property generates a PatternChange event.

Port Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPortC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPortX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPortA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPortS"}

Returns the name of the port through which a document is sent to a printer.

Syntax

object.**Port**

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

Remarks

The operating system determines the name of the port, such as LPT1: or LPT2:.

Note The effect of the properties of the **Printer** object depends on the driver supplied by the printer manufacturer. Some property settings may have no effect, or several different property settings may all have the same effect. Settings outside the accepted range may or may not produce an error. For more information, see the manufacturer's documentation for the specific driver.

PrevInstance Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPrevInstanceC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPrevInstanceX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPrevInstanceA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPrevInstanceS"}

Returns a value indicating whether a previous instance of an application is already running.

Syntax

object.**PrevInstance**

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

Remarks

You can use this property in a Load event procedure to specify whether a user is already running an instance of an application. Depending on the application, you might want only one instance running in the Microsoft Windows operating environment at a time.

PrintQuality Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPrintQualityC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPrintQualityX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPrintQualityA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPrintQualityS"}

Returns or sets a value indicating the printer resolution. Not available at design time.

Syntax

object.**PrintQuality** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A value or constant specifying printer resolution, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
vbPRPQDraft	-1	Draft resolution
vbPRPQLow	-2	Low resolution
vbPRPQMedium	-3	Medium resolution
vbPRPQHigh	-4	High resolution

In addition to the predefined negative values, you can also set *value* to a positive dots per inch (dpi) value, such as 300.

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

The default value depends on the printer driver and the current settings of the printer. The effect of these settings varies among printers and printer drivers. On some printers, some or all of the settings may produce the same result.

Note The effect of the properties of the **Printer** object depends on the driver supplied by the printer manufacturer. Some property settings may have no effect, or several different property settings may all have the same effect. Settings outside the accepted range may or may not produce an error. For more information, see the manufacturer's documentation for the specific driver.

RowHeight Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproRowHeightC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRowHeightX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRowHeightA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRowHeightS"}

For a **Grid** control, returns or sets the height of the specified row (record) in twips. Not available at design time.

For a **DBGrid** control, returns or sets the height of all rows in the control. **RowHeight** is always in the same unit of measure as the container for the **DBGrid** control.

Syntax

object.**RowHeight**[(*number*)] [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	The number of the row in a **Grid** control.
value	A numeric expression specifying height.

Remarks

For a **Grid** control, the minimum **RowHeight** is 1 pixel. Users can change the **RowHeight** of any row at run time by placing the mouse pointer on a gridline between rows and dragging.

ScaleHeight, ScaleWidth Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproScaleHeightC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproScaleHeightX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproScaleHeightA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproScaleHeightS"}

Return or set the number of units for the horizontal (**ScaleWidth**) and vertical (**ScaleHeight**) measurement of the interior of an object when using graphics methods or when positioning controls. For **MDIForm** objects, not available at design time and read-only at run time.

Syntax

object.**ScaleHeight** [**=** *value*]
object.**ScaleWidth** [**=** *value*]

The **ScaleHeight** and **ScaleWidth** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A numeric expression specifying the horizontal or vertical measurement.

Remarks

You can use these properties to create a custom coordinate scale for drawing or printing. For example, the statement `ScaleHeight = 100` changes the units of measure of the actual interior height of the form. Instead of the height being *n* current units (twips, pixels, ...), the height will be 100 user-defined units. Therefore, a distance of 50 units is half the height/width of the object, and a distance of 101 units will be off the object by 1 unit.

Use the **ScaleMode** property to define a scale based on a standard unit of measurement, such as twips, points, pixels, characters, inches, millimeters, or centimeters.

Setting these properties to positive values makes coordinates increase from top to bottom and left to right. Setting them to negative values makes coordinates increase from bottom to top and right to left.

Using these properties and the related **ScaleLeft** and **ScaleTop** properties, you can set up a full coordinate system with both positive and negative coordinates. All four of these Scale properties interact with the **ScaleMode** property in the following ways:

- Setting any other Scale property to any value automatically sets **ScaleMode** to 0. A **ScaleMode** of 0 is user-defined.
- Setting **ScaleMode** to a number greater than 0 changes **ScaleHeight** and **ScaleWidth** to the new unit of measurement and sets **ScaleLeft** and **ScaleTop** to 0. In addition, the **CurrentX** and **CurrentY** settings change to reflect the new coordinates of the current point.

You can also use the **Scale** method to set the **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, and **ScaleTop** properties in one statement.

Note The **ScaleHeight** and **ScaleWidth** properties aren't the same as the **Height** and **Width** properties.

For **MDIForm** objects, **ScaleHeight** and **ScaleWidth** refer only to the area not covered by **PictureBox** controls in the form. Avoid using these properties to size a **PictureBox** in the Resize event of an **MDIForm**.

ScaleLeft, ScaleTop Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproScaleLeftC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproScaleLeftX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproScaleLeftA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproScaleLeftS"}

Return or set the horizontal (**ScaleLeft**) and vertical (**ScaleTop**) coordinates for the left and top edges of an object when using graphics methods or when positioning controls.

Syntax

object.**ScaleLeft** [**=** *value*]
object.**ScaleTop** [**=** *value*]

The **ScaleLeft** and **ScaleTop** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A numeric expression specifying the horizontal or vertical coordinate. The default is 0.

Remarks

Using these properties and the related **ScaleHeight** and **ScaleWidth** properties, you can set up a full coordinate system with both positive and negative coordinates. These four Scale properties interact with the **ScaleMode** property in the following ways:

- Setting any other Scale property to any value automatically sets **ScaleMode** to 0. A **ScaleMode** of 0 is user-defined.
- Setting the **ScaleMode** property to a number greater than 0 changes **ScaleHeight** and **ScaleWidth** to the new unit of measurement and sets **ScaleLeft** and **ScaleTop** to 0. The **CurrentX** and **CurrentY** property settings change to reflect the new coordinates of the current point.

You can also use the **Scale** method to set the **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, and **ScaleTop** properties in one statement.

Note The **ScaleLeft** and **ScaleTop** properties aren't the same as the **Left** and **Top** properties.

ScaleMode Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproScaleModeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproScaleModeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproScaleModeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproScaleModeS"}

Returns or sets a value indicating the unit of measurement for coordinates of an object when using graphics methods or when positioning controls.

Syntax

object.**ScaleMode** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer specifying the unit of measurement, as described in Settings.

Settings

The settings for *value* are:

Constant	Setting	Description
vbUser	0	Indicates that one or more of the **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, and **ScaleTop** properties are set to custom values.
VbTwips	1	(Default) Twip (1440 twips per logical inch; 567 twips per logical centimeter).
VbPoints	2	Point (72 points per logical inch).
VbPixels	3	Pixel (smallest unit of monitor or printer resolution).
vbCharacters	4	Character (horizontal = 120 twips per unit; vertical = 240 twips per unit).
VbInches	5	Inch.
VbMillimeters	6	Millimeter.
VbCentimeters	7	Centimeter.

Remarks

Using the related **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, and **ScaleTop** properties, you can create a custom coordinate system with both positive and negative coordinates. These four Scale properties interact with the **ScaleMode** property in the following ways:

- Setting the value of any other Scale property to any value automatically sets **ScaleMode** to 0. A **ScaleMode** of 0 is user-defined.
- Setting the **ScaleMode** property to a number greater than 0 changes **ScaleHeight** and **ScaleWidth** to the new unit of measurement and sets **ScaleLeft** and **ScaleTop** to 0. The **CurrentX** and **CurrentY** property settings change to reflect the new coordinates of the current point.

ScrollBars Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproScrollBarsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproScrollBarsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproScrollBarsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproScrollBarsS"}

Returns or sets a value indicating whether an object has horizontal or vertical scroll bars. Read only at run time.

Syntax

object.**ScrollBars**

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

Settings

For an **MDIForm** object, the **ScrollBars** property settings are:

Setting	Description
True	(Default) The form has a horizontal or vertical scroll bar, or both.
False	The form has no scroll bars.

For a **Grid** or **TextBox** control, the **ScrollBars** property settings are:

Constant	Setting	Description
vbSBNone	0	(Default) None
vbHorizontal	1	Horizontal
vbVertical	2	Vertical
vbBoth	3	Both

Remarks

For a **TextBox** control with setting 1 (Horizontal), 2 (Vertical), or 3 (Both), you must set the **MultiLine** property to **True**.

At run time, the Microsoft Windows operating environment automatically implements a standard keyboard interface to allow navigation in **TextBox** controls with the arrow keys (UP ARROW, DOWN ARROW, LEFT ARROW, and RIGHT ARROW), the HOME and END keys, and so on.

Scroll bars are displayed on an object only if its contents extend beyond the object's borders. For example, in an **MDIForm** object, if part of a child form is hidden behind the border of the parent MDI form, a horizontal scroll bar (**HScrollBar** control) is displayed. Similarly, a vertical scroll bar (**VScrollBar** control) is displayed on a **Grid** control when it can't display all of its rows; a vertical scroll bar appears on a **TextBox** control when it can't display all of its lines of text. If **ScrollBars** is set to **False**, the object won't have scroll bars, regardless of its contents.

SelCount Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSelCountC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSelCountX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSelCountA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSelCountS"}

Returns the number of selected items in a **ListBox** control.

Syntax

object.**SelCount**

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

Remarks

The **SelCount** property returns 0 if no items are selected. Otherwise, it returns the number of list items currently selected. This property is particularly useful when users can make multiple selections.

Selected Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSelectedC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSelectedX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSelectedA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSelectedS"}

Returns or sets the selection status of an item in a **FileListBox** or **ListBox** control. This property is an array of Boolean values with the same number of items as the **List** property. Not available at design time.

Syntax

object.**Selected(***index***)** [**=** *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	The index number of the item in the control.
boolean	A Boolean expression specifying whether the item is selected, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The item is selected.
False	(Default) The item isn't selected.

Remarks

This property is particularly useful when users can make multiple selections. You can quickly check which items in a list are selected. You can also use this property to select or deselect items in a list from code.

If the **MultiSelect** property is set to 0, you can use the **ListIndex** property to get the index of the selected item. However, in a multiple selection, the **ListIndex** property returns the index of the item contained within the focus rectangle, whether or not the item is actually selected.

If a **ListBox** control's **Style** property is set to 1 (check boxes), the **Selected** property returns **True** only for those items whose check boxes are selected. The **Selected** property will not return **True** for those items which are only highlighted.

SelEndCol, SelStartCol, SelEndRow, SelStartRow Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSelEndColC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSelEndColX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSelEndColA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSelEndColS"}

Return or set the first or last row or column for a range of cells. Not available at design time.

- **SelEndCol** — the last selected column on the right.
- **SelStartCol** — the first selected column on the left.
- **SelEndRow** — the last selected row.
- **SelStartRow**—the first selected row.

Syntax

object.**SelEndCol** [**=** *value*]
object.**SelStartCol** [**=** *value*]
object.**SelEndRow** [**=** *value*]
object.**SelStartRow** [**=** *value*]

The **SelEndCol**, **SelStartCol**, **SelEndRow**, and **SelStartRow** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A numeric expression specifying the first or last column or row.

Remarks

You can use these properties to select a specific region of a **Grid** control from code or to return in code the dimensions of an area that the user selects.

SelStartCol and **SelStartRow** together specify the cell in the upper-left corner of a selected range. **SelEndCol** and **SelEndRow** specify the cell in the lower-right corner of a selected range.

To specify a cell without moving the current selection, use the **Col** and **Row** properties.

The default value for **SelStartCol** and **SelEndCol** is -1.

SelLength, SelStart, SelText Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSelLengthC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSelLengthX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSelLengthA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSelLengthS"}

- **SelLength** — returns or sets the number of characters selected.
- **SelStart** — returns or sets the starting point of text selected; indicates the position of the insertion point if no text is selected.
- **SelText** — returns or sets the string containing the currently selected text; consists of a zero-length string ("") if no characters are selected.

These properties aren't available at design time.

Syntax

object.**SelLength** [**=** *number*]
object.**SelStart** [**=** *index*]
object.**SelText** [**=** *value*]

The **SelLength**, **SelStart**, and **SelText** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression specifying the number of characters selected. For **SelLength** and **SelStart**, the valid range of settings is 0 to text length — the total number of characters in the edit area of a **ComboBox** or **TextBox** control.
index	A numeric expression specifying the starting point of the selected text, as described in Settings.
value	A string expression containing the selected text.

Remarks

Use these properties for tasks such as setting the insertion point, establishing an insertion range, selecting substrings in a control, or clearing text. Used in conjunction with the **Clipboard** object, these properties are useful for copy, cut, and paste operations.

When working with these properties:

- Setting **SelLength** less than 0 causes a run-time error.
- Setting **SelStart** greater than the text length sets the property to the existing text length; changing **SelStart** changes the selection to an insertion point and sets **SelLength** to 0.
- Setting **SelText** to a new value sets **SelLength** to 0 and replaces the selected text with the new string.

Shape Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproShapeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproShapeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproShapeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproShapeS"}

Returns or sets a value indicating the appearance of a **Shape** control.

Syntax

object.**Shape** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer specifying the control's appearance, as described in Settings.

Settings

The settings for *value* are:

Constant	Setting	Description
vbShapeRectangle	0	(Default) Rectangle
vbShapeSquare	1	Square
vbShapeOval	2	Oval
vbShapeCircle	3	Circle
VbShapeRoundedRectangle	4	Rounded Rectangle
VbShapeRoundedSquare	5	Rounded Square

Shortcut Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproShortcutC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproShortcutX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproShortcutA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproShortcutS"}

Sets a value that specifies a shortcut key for a **Menu** object. Not available at run time.

Remarks

Use this property to provide keyboard shortcuts for menu commands. You can set this property using the Menu Editor. For a list of shortcut keys you can use, see the Shortcut list in the Menu Editor.

Note In addition to shortcut keys, you can also assign access keys to commands, menus, and controls by using an ampersand (&) in the **Caption** property setting.

Stretch Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproStretchC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproStretchX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproStretchA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproStretchS"}

Returns or sets a value indicating whether a graphic resizes to fit the size of an **Image** control.

Syntax

object.**Stretch** [**=** *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying whether the graphic resizes, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The graphic resizes to fit the control.
False	(Default) The control resizes to fit the graphic.

Remarks

If **Stretch** is set to **True**, resizing the control also resizes the graphic it contains.

Style Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproStyleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproStyleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproStyleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproStyleS"}

Returns or sets a value indicating the type of combo box and the behavior of its list box portion. Read only at run time.

Syntax

object.**Style**

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

Settings

The **Style** property settings are:

Setting	Description
0	(Default) Dropdown Combo. Includes a drop-down list and a text box. The user can select from the list or type in the text box.
1	Simple Combo. Includes a text box and a list, which doesn't drop down. The user can select from the list or type in the text box. The size of a Simple combo box includes both the edit and list portions. By default, a Simple combo box is sized so that none of the list is displayed. Increase the **Height** property to display more of the list.
2	Dropdown List. This style only allows selection from the drop-down list.

Remarks

Follow these guidelines in deciding which setting to choose:

- Use setting 0 (Dropdown Combo) or setting 1 (Simple Combo) to give the user a list of choices. Either style enables the user to enter a choice in the text box. Setting 0 saves space on the form because the list portion closes when the user selects an item.
- Use setting 2 (Dropdown List) to display a fixed list of choices from which the user can select one. The list portion closes when the user selects an item.

TabStop Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTabStopC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTabStopX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTabStopA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTabStopS"}

Returns or sets a value indicating whether a user can use the TAB key to give the focus to an object.

Syntax

object.**TabStop** [**=** *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying whether the object is a tab stop, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) Designates the object as a tab stop.
False	Bypasses the object when the user is tabbing, although the object still holds its place in the actual tab order, as determined by the **TabIndex** property.

Remarks

This property enables you to add or remove a control from the tab order on a form. For example, if you're using a **PictureBox** control to draw a graphic, set its **TabStop** property to **False**, so the user can't tab to the **PictureBox**.

Title Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTitleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTitleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTitleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTitleS"}

Returns or sets the title of the application that is displayed in the Microsoft Windows Task List. If changed at run time, changes aren't saved with the application.

Syntax

object.**Title** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A string expression specifying the title of the application. The maximum length of *value* is 40 characters. In DBCS systems, this means the maximum length is 40 bytes.

Remarks

This property is available at design time in the dialog box for the Project Properties command on the Project menu.

TopIndex Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTopIndexC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTopIndexX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTopIndexA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTopIndexS"}

Returns or sets a value that specifies which item in a **FileListBox** or **ListBox** control is displayed in the topmost position. Not available at design time.

Syntax

object.**TopIndex** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	The number of the list item that is displayed in the topmost position. The default is 0, or the first item in the list.

Remarks

Use this property to scroll through a **FileListBox** or **ListBox** control without selecting an item.

If the **Columns** property is set to 0 for the **ListBox** control, the item is displayed at the topmost position if there are enough items below it to fill the visible portion of the list.

If the **Columns** property setting is greater than 0 for the **ListBox** control, the item's column moves to the leftmost position without changing its position within the column.

TopRow Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTopRowC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTopRowX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTopRowA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTopRowS"}

Returns or sets the topmost row displayed in a **Grid** control. Not available at design time.

Syntax

object.**TopRow** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	The number of the topmost row displayed.

Remarks

You can use this property in code to programmatically read or set the visible top row of the **Grid** control. Use the **LeftCol** property to determine the leftmost visible column in the **Grid**.

When setting this property, the largest possible row number is the total number of rows minus the number of rows that can be visible in the **Grid**. Attempting to set **TopRow** to a greater row number will generate an error.

TwipsPerPixelX, TwipsPerPixelY Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTwipsPerPixelXC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTwipsPerPixelXX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTwipsPerPixelXA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTwipsPerPixelXS"}

Return the number of twips per pixel for an object measured horizontally (**TwipsPerPixelX**) or vertically (**TwipsPerPixelY**).

Syntax

object.**TwipsPerPixelX**
object.**TwipsPerPixelY**

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

Remarks

Windows API routines generally require measurements in pixels. You can use these properties to convert measurements quickly without changing an object's **ScaleMode** property setting.

WindowList Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproWindowListC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproWindowListX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproWindowListA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproWindowListS"}

Returns or sets a value that determines whether a **Menu** object maintains a list of the current MDI child windows in an **MDIForm** object. Read only at run time.

Syntax

object.**WindowList**

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

Settings

The **WindowList** property settings are:

Setting	Description
True	The **Menu** object maintains a list of open windows and displays a check mark next to the active window. Users can click a window name to activate that window.
False	(Default) The **Menu** doesn't maintain a list of open windows.

Remarks

Many multiple-document interface (MDI) applications, such as Microsoft Excel and Microsoft Word for Windows, have a Window menu containing a list of open MDI child windows. This property enables you to add this functionality to your application.

Only one **Menu** object on a form can have its **WindowList** property set to **True**.

When you select the WindowList check box in the Menu Editor for a **Menu** object, the list of open MDI child windows for the menu you're creating is displayed.

WindowState Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproWindowStateC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproWindowStateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproWindowStateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproWindowStateS"}

Returns or sets a value indicating the visual state of a form window at run time.

Syntax

object.**WindowState** [**=** *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer specifying the state of the object, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
vbNormal	0	(Default) Normal.
VbMinimized	1	Minimized (minimized to an icon)
VbMaximized	2	Maximized (enlarged to maximum size)

Remarks

Before a form is displayed, the **WindowState** property is always set to Normal (0), regardless of its initial setting. This is reflected in the **Height**, **Left**, **ScaleHeight**, **ScaleWidth**, **Top**, and **Width** property settings. If a form is hidden after it's been shown, these properties reflect the previous state until the form is shown again, regardless of any changes made to the **WindowState** property in the meantime.

WordWrap Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproWordWrapC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproWordWrapX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproWordWrapA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproWordWrapS"}

Returns or sets a value indicating whether a **Label** control with its **AutoSize** property set to **True** expands vertically or horizontally to fit the text specified in its **Caption** property.

Syntax

object.**WordWrap** [**=** *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying whether the **Label** expands to fit the text, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The text wraps; the **Label** control expands or contracts vertically to fit the text and the size of the font. The horizontal size doesn't change.
False	(Default) The text doesn't wrap; the **Label** expands or contracts horizontally to fit the length of the text and vertically to fit the size of the font and the number of lines.

Remarks

Use this property to determine how a **Label** control displays its contents. For example, a graph that changes dynamically might have a **Label** containing text that also changes. To maintain a constant horizontal size for the **Label** and allow for increasing or decreasing text, set the **WordWrap** and **AutoSize** properties to **True**.

If you want a **Label** control to expand only horizontally, set **WordWrap** to **False**. If you don't want the **Label** to change size, set **AutoSize** to **False**.

Note If **AutoSize** is set to **False**, the text always wraps, regardless of the size of the **Label** control or the setting of the **WordWrap** property. This may obscure some text because the **Label** doesn't expand in any direction.

X1, Y1, X2, Y2 Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproX1C"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproX1X":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproX1A"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproX1S"}

Return or set the coordinates of the starting point (X1, Y1) and ending point (X2, Y2) of a **Line** control. The horizontal coordinates are X1 and X2; the vertical coordinates are Y1 and Y2.

Syntax

object.**X1** [**=** *value*]
object.**Y1** [**=** *value*]
object.**X2** [**=** *value*]
object.**Y2** [**=** *value*]

The **X1**, **Y1**, **X2**, and **Y2** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A numeric expression specifying a coordinate.

Remarks

Use these properties to dynamically extend a **Line** control from one point to another at run time. For example, you can show the relationships of items in one list to items in another list or connect points on a map.

MaxLength Property Example

This example uses a numeric value in one **TextBox** control to limit the length of text in another **TextBox** control. To try this example, paste the code into the Declarations section of a form that contains two **TextBox** controls. Make Text1 fairly large, and then press F5. Enter a number into Text2 and text into Text1.

```
Private Sub Text1_Change ()
   Text1.MaxLength = Text2.Text
End Sub
```

MDIChild Property Example

This example creates a second instance of an MDI child form within an **MDIForm** object. To try this example, set the **MDIChild** property to **True** on Form1, and then create an **MDIForm** object with the Add MDI Form command on the Project menu. Paste the code into the Declarations section of the **MDIForm**, and then press F5 to run the program.

```
Private Sub MDIForm_Load ()
   Dim NewForm As New Form1  ' Declare new form.
   NewForm.Show   ' Show new form.
End Sub
```

MouseIcon Property Example

This example illustrates how the **MouseIcon** property sets a custom mouse icon. To try the example, create a **ListBox** control on a form, and then set the **MultiSelect** property to 1 or 2. At run time, select one or more items. Different icons will appear, depending on whether you selected a single item or multiple items.

```
Private Sub Form_Load ()
   ' Put some items in the ListBox.
   List1.AddItem "Selection 1"
   List1.AddItem "Selection 2"
   List1.AddItem "Selection 3"
   List1.AddItem "Selection 4"
   List1.AddItem "Selection 5"
End Sub

Private Sub List1_MouseDown (Button As Integer, Shift As Integer, X As
Single, Y As Single)
   ' Set the custom mouse icon for multiple items.
   If List1.SelCount > 1 Then
      List1.MouseIcon = LoadPicture("ICONS\COMPUTER\MOUSE04.ICO")
      List1.MousePointer = 99
   Else  ' Set the custom mouse icon for a single item.
      List1.MouseIcon = LoadPicture("ICONS\COMPUTER\MOUSE02.ICO")
      List1.MousePointer = 99
   End If
End Sub
```

MousePointer Property Example

This example changes the mouse pointer to an hourglass while circles are drawn across the screen and then changes the hourglass back to a pointer at the end of the procedure. To try this example, paste the code into the Declarations section of a form. Press F5 to run the program, and then click the form.

```
Private Sub Form_Click ()
   Dim I ' Declare variable.
   ' Change mouse pointer to hourglass.
   Screen.MousePointer = vbHourglass
   ' Set random color and draw circles on form.
   For I = 0 To ScaleWidth Step 50
      ForeColor = RGB(Rnd * 255, Rnd * 255, Rnd * 255)
      Circle (I, ScaleHeight * Rnd), 400
   Next
   ' Return mouse pointer to normal.     Screen.MousePointer = vbDefault
End Sub
```

MultiSelect Property Example

This example fills a **ListBox** control with the names of your screen fonts and illustrates how the **MultiSelect** property affects the behavior of a **ListBox**. To try this example, create two **ListBox** controls and a **CommandButton** control on a form. In the first **ListBox**, set the **MultiSelect** property to 1 or 2. At run time, select several items in the first **ListBox**, and then click the **CommandButton**. All selected items are displayed in the second **ListBox**. Run the example several times with different settings of the **MultiSelect** property. Paste the code into the Declarations section, and then press F5 to run the program.

```
Private Sub Form_Load ()
   Dim I ' Declare variable.
   ' Fill the list box with screen font names.
   For I = 0 To Screen.FontCount - 1
      List1.AddItem Screen.Fonts(I)
   Next I
End Sub

Private Sub Command1_Click ()
   Dim I ' Declare variable.
   ' Clear all items from the list.
   List2.Clear
   ' If an item is selected, add it to List2.
   For I = 0 To List1.ListCount - 1
      If List1.Selected(I) Then
         List2.AddItem List1.List(I)
      End If
   Next I
End Sub
```

Page Property Example

This example prints three pages of text with the current page number at the top of each page. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim Header, I, Y  ' Declare variables.
   Print "Now printing..." ' Put notice on form.
   Header = "Printing Demo - Page "      ' Set header string.
   For I = 1 To 3
      Printer.Print Header;  ' Print header.
      Printer.Print Printer.Page   ' Print page number.
      Y = Printer.CurrentY + 10    ' Set position for line.
      ' Draw a line across page.
      Printer.Line (0, Y) - (Printer.ScaleWidth, Y)  ' Draw line.
      For K = 1 To 50
         Printer.Print String(K, " ");   ' Print string of spaces.
         Printer.Print "Visual Basic ";  ' Print text.
         Printer.Print Printer.Page      ' Print page number.
      Next
      Printer.NewPage
   Next I
   Printer.EndDoc
   End
End Sub
```

PasswordChar Property Example

This example illustrates how the **PasswordChar** property affects the way a **TextBox** control displays text. To try this example, paste the code into the Declarations section of a form that contains a **TextBox**, and then press F5 and click the form. Each time you click the form, the text toggles between an asterisk (*) password character and plain text.

```
Private Sub Form_Click ()
   If Text1.PasswordChar = "" Then
      Text1.PasswordChar = "*"
   Else
      Text1.PasswordChar = ""
   End If
End Sub
```

Pattern Property Example

This example updates a **TextBox** control with the new pattern selected in a **FileListBox** control. The controls are set up so that when the user enters a pattern in the **TextBox**, such as *.txt, it's reflected in the **FileListBox**, much like the interaction you see in a typical File Open dialog box in a Windows-based application. If a full path such as C:\Bin*.exe is entered into the **TextBox** control, the text is automatically parsed into path and pattern components by the **FileListBox** control. To try this example, paste the code into the Declarations section of a form that contains the following controls: a **DirListBox**, a **FileListBox**, a **TextBox**, and a **CommandButton**. Press F5 and type a valid file pattern into the **TextBox**.

```
Private Sub Form_Load ()
   Command1.Default = True ' Set Default property.
End Sub

Private Sub Command1_Click ()
   ' Text is parsed into path and pattern components.
   File1.Filename = Text1.Text
   Dir1.Path = File1.Path  ' Set directory path.
End Sub

Private Sub File1_PatternChange ()
   Text1.Text = File1.Pattern      ' Set text to new pattern.
End Sub

Private Sub Dir1_Change
   File1.Path = Dir1.Path  ' Set file list box path.
End Sub
```

Port Property Example

This example examines each **Printer** object in the **Printers** collection to find one connected to a specific port and makes it the default printer.

```
Dim P As Object
For Each P In Printers
   If P.Port = "LPT2:" Or P.DeviceName Like "*LaserJet*" Then
      Set Printer = P
      Exit For
   End If
Next P
```

RowHeight Property Example

This example sets the height of the current row to 500 twips when you click the form. To try this example, create a new project, use the Components dialog box to add a **Grid** control to the toolbox (from the Project menu, choose Components, and then select Microsoft Grid Control), and then draw a **Grid** control on the form. Paste the code into the Declarations section of the form, press F5 to run the program, and then select a cell and click the form.

```
Private Sub Form_Load ()
   Grid1.Rows = 5 ' Set columns and rows.
   Grid1.Cols = 7
End Sub

Private Sub Form_Click ()
   Grid1.RowHeight(Grid1.Row) = 500
End Sub
```

ScaleHeight, ScaleWidth Properties Example

This example uses the **ScaleHeight** and **ScaleWidth** properties to change the vertical and horizontal units of measurement for a form. To try this example, paste the code into the Declarations section of a form, and then press F5. To see the effect, click the form, resize it, and then click it again.

```
Private Sub Form_Click ()
   Dim Radius As Integer   ' Declare variable.
   ScaleHeight = 100 ' Set height units.
   ScaleWidth = 100  ' Set width units.
   For Radius = 5 to 50 Step 5
      FillStyle = 1
      Circle (50, 50), Radius      ' Draw circle.
   Next Radius
End Sub
```

ScaleLeft, ScaleTop Properties Example

This example creates a grid in a **PictureBox** control and sets coordinates for the upper-left corner to -1, -1 instead of 0, 0. Every 0.25 second, dots are randomly plotted from the upper-left corner to the lower-right corner. To try this example, paste the code into the Declarations section of a form that contains a large **PictureBox** and a **Timer** control, and then press F5.

```
Private Sub Form_Load ()
   Timer1.Interval = 250   ' Set Timer interval.
   Picture1.ScaleTop = -1  ' Set scale for top of grid.
   Picture1.ScaleLeft = -1 ' Set scale for left of grid.
   Picture1.ScaleWidth = 2 ' Set scale (-1 to 1).
   Picture1.ScaleHeight = 2
   Picture1.Line (-1, 0)-(1, 0)    ' Draw horizontal line.
   Picture1.Line (0, -1)-(0, 1)    ' Draw vertical line.
End Sub

Private Sub Timer1_Timer ()
   Dim I ' Declare variable.
   ' Plot dots randomly within a range.
   For I = -1 To 1 Step .05
      Picture1.PSet (I * Rnd, I * Rnd)   ' Draw a point.
   Next I
End Sub
```

ScaleMode Property Example

This example shows how different **ScaleMode** property settings change the size of a circle. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form. When you click the form, the unit of measurement changes to the next **ScaleMode** setting and a circle is drawn on the form.

```
Private Sub Form_Click ()
   ' Cycle through each of the seven ScaleMode settings.
   ScaleMode = ((ScaleMode + 1) Mod 7) + 1
   ' Draw a circle with radius of 2 in center of form.
   Circle (ScaleWidth / 2, ScaleHeight / 2), 2
End Sub
```

SelLength, SelStart, SelText Properties Example

This example enables the user to specify some text to search for and then searches for the text and selects it, if found. To try this example, paste the code into the Declarations section of a form that contains a wide **TextBox** control, and then press F5 and click the form.

```
Private Sub Form_Load ()
   Text1.Text = "Two of the peak human experiences"
   Text1.Text = Text1.Text & " are good food and classical music."
End Sub
Private Sub Form_Click ()
   Dim Search, Where ' Declare variables.
   ' Get search string from user.
   Search = InputBox("Enter text to be found:")
   Where = InStr(Text1.Text, Search)     ' Find string in text.
   If Where Then  ' If found,
      Text1.SelStart = Where - 1   ' set selection start and
      Text1.SelLength = Len(Search)      ' set selection length.
   Else
      MsgBox "String not found."   ' Notify user.
   End If
End Sub
```

This example shows how the **Clipboard** object is used in cut, copy, paste, and delete operations. To try this example, create a form with a **TextBox** control and use the Menu Editor to create an Edit menu (for each of the commands, set the **Caption** property = Cut, Copy, Paste, and Delete, respectively; set the **Name** property = EditCut, EditCopy, EditPaste, and EditDelete, respectively).

```
Private Sub EditCut_Click ()
   ' Clear the contents of the Clipboard.
   Clipboard.Clear
   ' Copy selected text to Clipboard.
   ClipBoard.SetText Screen.ActiveControl.SelText
   ' Delete selected text.
   Screen.ActiveControl.SelText = ""
End Sub

Private Sub EditCopy_Click ()
   ' Clear the contents of the Clipboard.
   Clipboard.Clear
   ' Copy selected text to Clipboard.
   ClipBoard.SetText Screen.ActiveControl.SelText
End Sub

Private Sub EditPaste_Click ()
   ' Place text from Clipboard into active control.
   Screen.ActiveControl.SelText = ClipBoard.GetText ()
End Sub

Private Sub EditDelete_Click ()
' Delete selected text.
Screen.ActiveControl.SelText = ""
End Sub
```

Shape Property Example

This example illustrates the six possible shapes of the **Shape** control. To try this example, paste the code into the Declarations section of a form that contains an **OptionButton** control and a **Shape** control. For the **OptionButton**, set the **Index** property to 0 to create a control array of one element, and then press F5. Click each **OptionButton** to see each different shape.

```
Private Sub Form_Load ()
   Dim I ' Declare variable.
   Option1(0).Caption = "Shape #0"
   For I = 1 To 5    ' Create five instances of Option1.
      Load Option1(I)
      ' Set the location of the new option button.
      Option1(I).Top = Option1(I - 1).Top + Option1(0).Height + 40
      ' Set the option button caption.
      Option1(I).Caption = "Shape #" & I
      ' Display the new option button.
      Option1(I).Visible = True
   Next I
End Sub

Private Sub Option1_Click (Index As Integer)
   Shape1.Shape = Index
End Sub
```

Stretch Property Example

This example loads an arrow icon from an icons directory into an **Image** control. The arrow crawls across the form when the **Stretch** property is set to **True** and hops across the form when **Stretch** is set to **False**. To try this example, paste the code into the Declarations section of a form that contains an **Image** control, a **CheckBox** control, and a **Timer** control, and then press F5 and click the form. Be sure to check the path to your icons directory and change it if necessary. To see the effects of the **Stretch** property, click the **CheckBox**, and then click the form again.

```
Dim ImgW ' Declare variable.
Private Sub Form_Load ()
   ' Load an icon into the Image control.
   Image1.Picture = LoadPicture("ICONS\ARROWS\ARW02RT.ICO")
   Image1.Left = 0   ' Move image to left edge.
   ImgW = Image1.Width  ' Save width of image.
   Timer1.Interval = 300
   Timer1.Enabled = False  ' Turn off timer.
   Check1.Caption = "Stretch Property"
End Sub

Private Sub Form_Click ()
   Timer1.Enabled = True   ' Turn on the timer.
End Sub

Private Sub Timer1_Timer ()
   Static MoveIcon As Integer      ' Flag for moving the icon.
   If Not MoveIcon Then
      Image1.Move Image1.Left + ImgW, Image1.Top, ImgW * 2
   Else
      ' Move the image and return it to original width.
      Image1.Move Image1.Left + ImgW, Image1.Top, ImgW
   End If
   ' If image is off edge of form, start over.
   If Image1.Left > ScaleWidth Then
      Image1.Left = 0
      Timer1.Enabled = False
   End If
   MoveIcon = Not MoveIcon ' Reset flag.
End Sub

Private Sub Check1_Click ()
   Image1.Stretch = Check1.Value
End Sub
```

TopIndex Property Example

This example fills a **ListBox** control with names of screen fonts and then scrolls through the **ListBox** when you click the form. To try this example, paste the code into the Declarations section of a form that contains a **ListBox** control, and then press F5 and click the form.

```
Private Sub Form_Load ()
   Dim I ' Declare variable.
   For I = 0 To Screen.FontCount -1      ' Fill list box with
      List1.AddItem Screen.Fonts(I)      ' screen font names.
   Next I
End Sub

Private Sub Form_Click ()
   Dim X ' Declare variable.
   X = List1.TopIndex    ' Get current index.
   List1.TopIndex = List1.TopIndex + 5    ' Reset topmost item.
   If List1.TopIndex = X Then List1.TopIndex = 0
End Sub
```

TopRow Property Example

This example changes the **TopRow** property setting with each click on the form. To try this example, create a new project, use the Components dialog box to add a **Grid** control to the toolbox (from the Project menu, choose Components, and then select Microsoft Grid Control), and then draw a **Grid** control. Paste the code into the Declarations section of the form, and then press F5 to run the program.

```
Private Sub Form_Load ()
   ' Set columns and rows.
   Grid1.Cols = 6
   Grid1.Rows = 12
   ' Put numbers in row heads.
   Grid1.Col = 0
   For I = 1 To Grid1.Rows - 1
      Grid1.Row = I
      Grid1.Text = I
   Next I
End Sub

Private Sub Form_Click ()
   On Error GoTo TopRowError
   Grid1.TopRow = Grid1.TopRow + 1
   On Error GoTo 0
   Exit Sub

TopRowError:
   Grid1.TopRow = 1
   Resume Next
End Sub
```

WindowList Property Example

This example creates some menu commands, illustrates the **WindowList** menu functionality, and shows how to enable your users to add new forms to a multiple-document interface (MDI) application. To try this example, create an **MDIForm** object with the Add MDI Form command on the Project menu. On Form1, set the **MDIChild** property to **True**, and create a menu named File. Select the WindowList box for the File menu. On your File menu, create a New command, set its **Name** property to FileMenu, and set its **Index** property to 0 to create a control array. Paste the code into the Declarations section of the form, and then press F5 to run the program. Choosing the New command on the File menu creates new MDI child forms. Their names are listed at the bottom of the File menu.

```
Private Sub Form_Load ()
   FileMenu(0).Caption = "&New"    ' Set access key in caption.
   Load FileMenu(1)  ' Create new menu item.
   FileMenu(1).Caption = "-" ' Set separator.
   Load FileMenu(2)  ' Create new menu item.
   FileMenu(2).Caption = "E&xit"   ' Set caption and access key.
End Sub

Private Sub FileMenu_Click (Index As Integer)
   Select Case Index
      Case 0   ' Select New command.
         Dim NewForm As New Form1  ' Create a duplicate of Form1.
         ' Load NewForm and set a unique caption.
         NewForm.Caption = "Untitled" & Forms.Count
      Case 2   ' Select Exit command.
         End   ' End the program.
   End Select
End Sub
```

WindowState Property Example

This example hides a dialog box (Form2) when the parent form (Form1) is minimized and redisplays the dialog box when the parent form is returned to either an original or maximized state. To try this example, paste the code into the Declarations section of Form1 of an application that contains two forms. Press F5 to start the example. Move Form1 so you can see both forms, and then minimize or maximize the form and observe the behavior of Form2.

```
Private Sub Form_Load ()
   Form2.Show  ' Show Form2.
End Sub

Private Sub Form_Resize ()
   ' If parent form is minimized...
   If Form1.WindowState = vbMinimized Then
   ' ...hide Form2.
      Form2.Visible = False
   ' If parent form isn't minimized...
   Else
      ' ...restore Form2.
      Form2.Visible = True
   End If
End Sub
```

WordWrap Property Example

This example puts text into two **Label** controls and uses the **WordWrap** property to illustrate their different behavior. To try this example, paste the code into the Declarations section of a form that contains two **Label** controls, and then press F5 and click the form to toggle the **WordWrap** property setting.

```
Private Sub Form_Load ()
   Dim Author1, Author2, Quote1, Quote2  ' Declare variables.
   Label1.AutoSize = True  ' Set AutoSize.
   Label2.AutoSize = True
   Label1.WordWrap = True  ' Set WordWrap.
   Quote1 = "I couldn't wait for success, so I went on without it."
   Author1 = "  - Jonathan Winters"
   Quote2 = "Logic is a system whereby one may go wrong with confidence."
   Author2 = "  - Charles Kettering"
   Label1.Caption = Quote1 & Chr(10) & Author1
   Label2.Caption = Quote2 & Chr(10) & Author2
End Sub

Private Sub Form_Click ()
   Label1.Width = 1440  ' Set width to 1 inch in twips.
   Label2.Width = 1440
   Label1.WordWrap = Not Label1.WordWrap ' Toggle WordWrap property.
   Label2.WordWrap = Not Label2.WordWrap
End Sub
```

X1, Y1, X2, Y2 Properties Example

This example displays an animated line that walks down the form when you click the form. To try this example, paste the code into the Declarations section of a form that contains a **Timer** control and a **Line** control, and then press F5 and click the form.

```
Private Sub Form_Load ()
   Timer1.Interval = 100   ' Set Timer interval.
   ' Position the line near the upper-left corner.
   ' Set Line1's properties.
   With Line1
      .X1 = 100
      .Y1 = 100
      .X2 = 500
      .Y2 = 300
   End With
   Timer1.Enabled = False
End Sub

Private Sub Form_Click ()
   Timer1.Enabled = True   ' Start the timer.
End Sub

Private Sub Timer1_Timer ()
   Static Odd  ' Declare variable.
   If Odd Then
      Line1.X2 = Line1.X2 + 250
      Line1.Y2 = Line1.Y2 + 600
   Else
      Line1.X1 = Line1.X1 + 250
      Line1.Y1 = Line1.Y1 + 600
   End If
   Odd = Not Odd  ' Toggle the value.
   ' If the line is off the form, start over.
   If Line1.Y1 > ScaleHeight Then
      Timer1.Enabled = False ' Wait for another click.
      With Line1
         .X1 = 100
         .Y1 = 100
         .X2 = 500
         .Y2 = 300
      End With
      Odd = False
   End If
End Sub
```

ActiveControl Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproActiveControlC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproActiveControlX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproActiveControlA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproActiveControlS"}

Returns the control that has the focus. When a form is referenced, as in `ChildForm.ActiveControl`, **ActiveControl** specifies the control that would have the focus if the referenced form were active. Not available at design time; read-only at run time.

Syntax

object.**ActiveControl**

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

Remarks

You can use **ActiveControl** to access a control's properties or to invoke its methods: For example, `Screen.ActiveControl.Tag = "0"`. A run-time error occurs if all controls on the form are invisible or disabled.

Each form can have an active control (`Form.ActiveControl`), regardless of whether or not the form is active. You can write code that manipulates the active control on each form in your application even when the form isn't the active form.

This property is especially useful in a multiple-document interface (MDI) application where a button on a toolbar must initiate an action on a control in an MDI child form. When a user clicks the Copy button on the toolbar, your code can reference the text in the active control on the MDI child form, as in `ActiveForm.ActiveControl.SelText`.

Note If you plan to pass `Screen.ActiveControl` to a procedure, you must declare the argument in that procedure with the clause `As Control` rather than specifying a control type (`As TextBox` or `As ListBox`) even if `ActiveControl` always refers to the same type of control.

Appearance Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAuto3DC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAuto3DX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAuto3DA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAuto3DS"}

Returns or sets the paint style of controls on an **MDIForm** or **Form** object at run time. Read-only at run time.

Syntax

object.**Appearance**

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

Settings

The **Appearance** property settings are:

Setting	Description
0	Flat. Paints controls and forms without visual effects.
1	(Default) 3D. Paints controls with three-dimensional effects.

Remarks

If set to 1at design time, the **Appearance** property draws controls with three-dimensional effects. If the form's **BorderStyle** property is set to Fixed Double (**vbFixedDouble**, or 3), the caption and border of the form are also painted with three-dimensional effects. Setting the **Appearance** property to 1 also causes the form and its controls to have their **BackColor** property set to the color selected for Button Face in the Color option of the operating system's Control Panel.

Setting the **Appearance** property to 1 for an **MDIForm** object affects only the MDI parent form. To have three-dimensional effects on MDI child forms, you must set each child form's **Appearance** property to 1.

Bold Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBoldC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBoldX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBoldA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBoldS"}

Returns or sets the font style of the **Font** object to either bold or nonbold.

Syntax

object.**Bold** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying the font style, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	Turns on bold formatting.
False	(Default) Turns off bold formatting.

Remarks

The **Font** object isn't directly available at design time. Instead you set the **Bold** property by selecting a control's **Font** property in the Properties window and clicking the Properties button. In the Font Style box of the Font dialog box, select either Bold or Bold Italic. At run time, however, you set **Bold** directly by specifying its setting for the **Font** object.

FontTransparent Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFontTransparentC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontTransparentX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontTransparentA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontTransparentS"}

Returns or sets a value that determines whether background text and graphics on a **Form** or **Printer** object or a **PictureBox** control are displayed in the spaces around characters.

Syntax

object.**FontTransparent** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying the state of background text and graphics, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) Permits background graphics and text to show around the spaces of the characters in a font.
False	Masks existing background graphics and text around the characters of a font.

Remarks

Set **FontTransparent** at design time using the Properties window or at run time using code. Changing **FontTransparent** at run time doesn't affect graphics and text already drawn to **Form**, **Printer**, or **PictureBox**.

Italic Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproItalicC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBoldX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproItalicA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproItalicS"}

Returns or sets the font style of the **Font** object to either italic or nonitalic.

Syntax

object.**Italic** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying the font style as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	Turns on italic formatting.
False	(Default) Turns off italic formatting.

Remarks

The **Font** object isn't directly available at design time. Instead you set the **Italic** property by selecting a control's **Font** property in the Properties window and clicking the Properties button. In the Font Style box of the Font dialog box, select either Italic or Bold Italic. At run time, however, you set **Italic** directly by specifying its setting for the **Font** object.

LBound Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLBoundC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLBoundX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLBoundA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLBoundS"}

Returns the lowest ordinal value of a control in a control array.

Syntax

object.**LBound**

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

Remarks

The **LBound** property setting is equal to the **Index** property value of the first control in the array. Typically this value is 0 because Visual Basic automatically assigns an **Index** value of 0 to the first control in a control array. If you manually change the **Index** value for the first control in an array to some other value (for example, 1), **LBound** returns the value you manually assigned to **Index** (in this example, 1).

Size Property (Font)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSizeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBoldX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSizeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSizeS"}

Returns or sets the font size used in the **Font** object.

Syntax

object.**Size** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression specifying the size of the font in points.

Remarks

Use this property to format text in the font size you want. The default font size is determined by the operating system. To change the default, specify the size of the font in points. The maximum value for the **Size** property is 2048 points.

The **Font** object isn't directly available at design time. Instead you set the **Size** property by selecting a control's **Font** property in the Properties window and clicking the Properties button. In the Size box of the Font dialog box, select the size you want. At run time, however, set **Size** directly by specifying its setting for the **Font** object.

StartMode Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproStartModeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproStartModeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproStartModeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproStartModeS"}

Returns or sets a value that determines whether an application starts as a stand-alone project or as an ActiveX component. Read-only at run time.

Syntax

object.**StartMode**

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

Settings

The **StartMode** property settings are:

Constant	Value	Description
vbSModeStandalone	0	(Default) Application starts as a stand-alone project.
VbSModeAutomation	1	Application starts as an ActiveX component.

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

At design time, you can set **StartMode** in the Project Options dialog box to 1 (**vbSModeAutomation**) to debug an application as if it were started as an ActiveX component.

Once a project is compiled, the value of the **StartMode** property is determined by how that application is started, not by its nominal setting in the Project Options dialog box.

When **StartMode** is set to 1 and there are no public classes in the project, you must use the **End** statement or choose End from the Run menu or toolbar to end the application. If you choose Close from the System menu, the form closes but the project is still running.

StrikeThrough Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproStrikeThroughC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBoldX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproStrikeThroughA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproStrikeThroughS"}

Returns or sets the font style of the **Font** object to either strikethrough or nonstrikethrough.

Syntax

object.**StrikeThrough** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying the font style, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	Turns on strikethrough formatting.
False	(Default) Turns off strikethrough formatting.

Remarks

The **Font** object isn't directly available at design time. Instead you set the **StrikeThrough** property by choosing a control's **Font** property in the Properties window and clicking the Properties button. In the Font dialog box, select the Strikeout check box. At run time, however, you set **StrikeThrough** directly by specifying its setting for the **Font** object.

TrackDefault Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTrackDefaultC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTrackDefaultX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTrackDefaultA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTrackDefaultS"}

Returns or sets a value that determines whether the **Printer** object always points to the same printer or changes the printer it points to if you change the default printer setting in the operating system's Control Panel. Not available at design time.

Syntax

object.**TrackDefault** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying the printer *object* points to, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) The **Printer** object changes the printer it points to when you change the default printer settings in the operating system's Control Panel.
False	The **Printer** object continues to point to the same printer even though you change the default printer settings in the operating system's Control Panel.

Remarks

Changing the **TrackDefault** property setting while a print job is in progress sends an implicit **EndPage** statement to the **Printer** object.

Type Property (Picture)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTypeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTypeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTypeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTypeS"}

Returns the graphic format of a **Picture** object. Not available at design time; read-only at run time.

Syntax

object.**Type**

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

Return Values

The return values for the **Type** property are:

Constant	Value	Description
vbPicTypeBitmap	1	Bitmap (.bmp files)
vbPicTypeMetafile	2	Metafile (.wmf files)
vbPicTypeIcon	3	Icon (.ico files)
vbPicTypeNone	0	Picture is empty
vbPicTypeEMetafile	3	Enhanced Metafile (.emf files)

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

UBound Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproUBoundC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLBoundX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproUBoundA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproUBoundS"}

Returns the highest ordinal value of a control in a control array.

Syntax

object.**UBound**

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

Remarks

UBound is equal to the **Index** property value of the last control in the array.

Underline Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproUnderlineC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBoldX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproUnderlineA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproUnderlineS"}

Returns or sets the font style of the **Font** object to either underlined or nonunderlined.

Syntax

object.**Underline** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying the font style, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	Turns on underline formatting.
False	(Default) Turns off underline formatting.

Remarks

The **Font** object isn't directly available at design time. Instead you set the **Underline** property by selecting a control's **Font** property in the Properties window and clicking the Properties button. In the Font dialog box, select the Underline check box. At run time, however, you set **Underline** directly by specifying its setting for the **Font** object.

UseMnemonic Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproUsemnemonicC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproUseMnemonicX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproUsemnemonicA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproUseMnemonicS"}

Returns or sets a value that specifies whether an ampersand (&) included in the text of the **Caption** property of the **Label** control defines an access key.

Syntax

object.**UseMnemonic** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying whether the **Label** control enables an access key, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) Any ampersand appearing in the text of the **Caption** property causes the character following the ampersand to become an access key. The ampersand itself isn't displayed in the interface of the **Label** control.
False	Any ampersand appearing in the text of the **Caption** property is displayed as an ampersand in the interface of the **Label** control.

Remarks

At run time, pressing ALT+ the access key defined in the **Label** control's **Caption** property moves focus to the control that follows the **Label** control in the tab order.

Weight Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproWeightC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBoldX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproWeightA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproWeightS"}

Returns or sets the weight of the characters that make up a **Font** object. The weight refers to the thickness of the characters, or the "boldness factor". The higher the value, the bolder the character.

Syntax

object.**Weight** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression specifying the weight of the font.

Remarks

The **Font** object isn't directly available at design time. You set the **Weight** property of the **Font** object by selecting a control's **Font** property in the Properties window and clicking the Properties button. You implicitly set the **Weight** property by selecting an item from the Font Style box in the Font dialog box. The Regular and Italic settings have a **Weight** value of 400 (the default), and the Bold and Bold Italic settings have a **Weight** value of 700. At run time, however, you set **Weight** directly by specifying its setting for the **Font** object.

If you set a **Font** object's **Weight** to a value other than 400 or 700 at run time, Visual Basic converts your value to either 400 or 700, depending on which value is closest to the value you set. The precise ranges are: **Weight** > 400 and < 551 converts to 400; **Weight** > 550 converts to 700.

ActiveControl Property Example

This example displays the text of the active control. To try this example, paste the code into the Declarations section of a form that contains **TextBox**, **Label**, and **CommandButton** controls, and then press F5 and click the form.

```
Private Sub Form_Click ()
   If TypeOf Screen.ActiveControl Is TextBox Then
      Label1.Caption = Screen.ActiveControl.Text
   Else
      Label1.Caption = "Button: " + Screen.ActiveControl.Caption
   End If
End Sub
```

This example shows how you can use the **Clipboard** object in cut, copy, paste, and delete operations using buttons on a toolbar. To try this example, put **TextBox** and **CheckBox** controls on Form1, and then create a new MDI form. On the MDI form, insert a **PictureBox** control, and then insert a **CommandButton** in the **PictureBox**. Set the **Index** property of the **CommandButton** to 0 (creating a control array). Set the **MDIChild** property of Form1 to **True**.

To run the example, copy the code into the Declarations section of the **MDIForm**, and then press F5. Notice that when the **CheckBox** has the focus, the buttons don't work, since the **CheckBox** is now the active control instead of the **TextBox**.

```
Private Sub MDIForm_Load ()
   Dim I ' Declare variable.
   Command1(0).Move 0, 0, 700, 300 ' Position button on toolbar.
   For I = 1 To 3 ' Create other buttons.
      Load Command1(I)   ' Create button.
      Command1(I).Move I * 700, 0, 700, 300   ' Place and size button.
      Command1(I).Visible = True   ' Display button.
   Next I
   Command1(0).Caption = "Cut"     ' Set button captions.
   Command1(1).Caption = "Copy"
   Command1(2).Caption = "Paste"
   Command1(3).Caption = "Del"
End Sub

Private Sub Command1_Click (Index As Integer)
   ' ActiveForm refers to the active form in the MDI form.
   If TypeOf ActiveForm.ActiveControl Is TextBox Then
      Select Case Index
         Case 0   ' Cut.
            ' Copy selected text onto Clipboard.
            Clipboard.SetText ActiveForm.ActiveControl.SelText
            ' Delete selected text.
            ActiveForm.ActiveControl.SelText = ""
         Case 1   ' Copy.
            ' Copy selected text onto Clipboard.
            Clipboard.SetText ActiveForm.ActiveControl.SelText
         Case 2   ' Paste.
            ' Put Clipboard text in text box.
            ActiveForm.ActiveControl.SelText = Clipboard.GetText()
         Case 3   ' Delete.
            ' Delete selected text.
            ActiveForm.ActiveControl.SelText = ""
      End Select
   End If
```

```
End Sub
```

Bold, Italic, Size, StrikeThrough, Underline, Weight Properties Example

This example prints text on a form with each mouse click. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form twice.

```
Private Sub Form_Click ()
   Font.Bold = Not Font.Bold ' Toggle bold.
   Font.StrikeThrough = Not Font.StrikeThrough ' Toggle strikethrough.
   Font.Italic = Not Font.Italic    ' Toggle italic.
   Font.Underline = Not Font.Underline    ' Toggle underline.
   Font.Size = 16 ' Set Size property.
   If Font.Bold Then
      Print "Font weight is " & Font.Weight & " (bold)."
   Else
      Print "Font weight is " & Font.Weight & " (not bold)."
   End If
End Sub
```

FontTransparent Property Example

This example prints text on top of a graphic in a **PictureBox** control. Put a **PictureBox** on a form, set its **AutoSize** property to **True**, and load its **Picture** property with a bitmap (.bmp) file. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form twice.

```
Private Sub Form_Click ()
   ' Toggle property.
   Picture1.FontTransparent = Not Picture1.FontTransparent
   Picture1.Print "Demo of FontTransparent property."
End Sub
```

LBound, UBound Properties Example

This example prints the values of these two properties for a control array. Put an **OptionButton** control on a form, and set its **Index** property to 0 (to create a control array). To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Paint ()
   Static FlagFormPainted As Integer
   If FlagFormPainted <> True Then ' When form is painting for first time,
      For i = 1 To 3
         Load Option1(i)   ' add three option buttons to array.
         Option1(i).Top = Option1(i - 1).Top + 350
         Option1(i).Visible = True
      Next I
      For I =  0 to 3   ' Put captions on the option buttons.
         Option1(i).Caption = "Option #" & CStr(i)
      Next I
      Option1(0).Value = True      ' Select first option button.
      FlagFormPainted = True ' Form is done painting.
   End If
End Sub
Private Sub Form_Click ()
   Print "Control array's Count property is " & Option1().Count
   Print "Control array's LBound property is " & Option1().LBound
   Print "Control array's UBound property is " & Option1().UBound
End Sub
```

StartMode Property Example

This example shows one possible effect of setting the **StartMode** property to 1 (**vbSModeAutomation**) at design time. Create an ActiveX EXE project. Create a new form. From the Project menu, choose the Project Properties command. Select the Component tab, then select the ActiveX component option button in the Start Mode group. Choose OK to close the Options dialog box. To try this example, paste the code into the Declarations section of the form, and then press F5 and double-click the Control menu at the left of the form's title bar. If the form doesn't display, enter `Form1.Show` in the Immediate window.

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
   If UnloadMode = vbFormControlMenu And App.StartMode = vbSModeAutomation
Then
      Msg = "Form will close but application will still be running." &
Chr(10)
      Msg = Msg + "To terminate application without a public class," &
Chr(10)
      Msg = Msg + "you must use an End statement."
      MsgBox Msg
   End If
End Sub
```

UseMnemonic Property Example

This example reads the setting of the **UseMnemonic** property of a **Label** control. To try this example, paste the code into the Declarations section of a form that contains a **Label**, and then press F5 and click the form.

```
Private Sub Form_Click()
   If Label1.UseMnemonic And InStr(Label1, "&") Then
      MsgBox "The label has an access key character."
   ElseIf Label1.UseMnemonic And Not InStr(Label1, "&") Then
      MsgBox "The label supports an access key character but doesn't have
an ampersand."
   Else
      MsgBox "The label doesn't support an access key character."
   End If
End Sub
```

Type, Width Properties Example

This example reads the setting of the **Type** and **Width** properties of a **Picture** object in a **PictureBox** control. To try this example, paste the code into the Declarations section of a form that contains a **PictureBox** whose **Picture** property is set to an icon, and then press F5 and click the form.

```
Private Sub Form_Click()
   If Picture1.Picture.Type = vbPicTypeIcon Then
      Print "The graphic in the picture box is an icon."
   Else
      Print "The Picture property isn't set to an icon."
   End If
   Print "Width of the graphic in HiMetrics is " & Picture1.Picture.Width
   Print "Width of picture box itself in twips is " & Picture1.Width
End Sub
```

ActiveForm Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproActiveFormC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproActiveFormX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproActiveFormA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproActiveFormS"}

Returns the form that is the active window. If an **MDIForm** object is active or is referenced, it specifies the active MDI child form.

Syntax

object.**ActiveForm**

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

Remarks

Use the **ActiveForm** property to access a form's properties or to invoke its methods — for example, `Screen.ActiveForm.MousePointer = 4`.

This property is especially useful in a multiple-document interface (MDI) application where a button on a toolbar must initiate an action on a control in an MDI child form. When a user clicks the Copy button on the toolbar, your code can reference the text in the active control on the MDI child form — for example, `ActiveForm.ActiveControl.SelText`.

When a control on a form has the focus, that form is the active form on the screen (`Screen.ActiveForm`). In addition, an **MDIForm** object can contain one child form that is the active form within the context of the MDI parent form (`MDIForm.ActiveForm`). The **ActiveForm** on the screen isn't necessarily the same as the **ActiveForm** in the MDI form, such as when a dialog box is active. For this reason, specify the **MDIForm** with **ActiveForm** when there is a chance of a dialog box being the **ActiveForm** property setting.

Note When an active MDI child form isn't maximized, the title bars of both the parent form and the child form appear active.

If you plan to pass `Screen.ActiveForm` or `MDIForm.ActiveForm` to a procedure, you must declare the argument in that procedure with the generic type (`As Form`) rather than a specific form type (`As MyForm`) even if **ActiveForm** always refers to the same type of form.

The **ActiveForm** property determines the default value for the **ProjectTemplate** object.

Caption Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproCaptionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCaptionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCaptionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCaptionS"}

- Form — determines the text displayed in the **Form** or **MDIForm** object's title bar . When the form is minimized, this text is displayed below the form's icon.
- Control — determines the text displayed in or next to a control.
- **MenuLine** object — determines the text displayed for a **Menu** control or an object in the **MenuItems** collection.

For a **Menu** control, **Caption** is normally read/write at run time. But **Caption** is read-only for menus that are exposed or supplied by Visual Basic to add-ins, such as the **MenuLine** object.

Syntax

object.**Caption** [= *string*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the form associated with the active form moduleis assumed to be *object.*
string	A string expression that evaluates to the text displayed as the caption.

Remarks

When you create a new object, its default caption is the default **Name** property setting. This default caption includes the object name and an integer, such as Command1 or Form1. For a more descriptive label, set the **Caption** property.

You can use the **Caption** property to assign an access key to a control. In the caption, include an ampersand (&) immediately preceding the character you want to designate as an access key. The character is underlined. Press the ALT key plus the underlined character to move the focus to that control. To include an ampersand in a caption without creating an access key, include two ampersands (&&). A single ampersand is displayed in the caption and no characters are underlined.

A **Label** control's caption size is unlimited. For forms and all other controls that have captions, the limit is 255 characters.

To display the caption for a form, set the **BorderStyle** property to either Fixed Single (1 or **vbFixedSingle**), Sizable (2 or **vbSizable**), or Fixed Double (3 or **vbFixedDouble**). A caption too long for the form's title bar is clipped. When an MDI child form is maximized within an **MDIForm** object, the child form's caption is included in the parent form's caption.

Tip For a label, set the **AutoSize** property to **True** to automatically resize the control to fit its caption.

Checked Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproCheckedC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCheckedX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCheckedA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCheckedS"}

Returns or sets a value that determines whether a check mark is displayed next to a menu item.

Syntax

object.**Checked** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that specifies whether a check mark is displayed next to a menu item.

Settings

The settings for *boolean* are:

Setting	Description
True	Places a check mark next to a menu item.
False	(Default) Doesn't place a check mark next to a menu item.

Remarks

At design time, you can use the Menu Editor to set **Checked** to **True**. At run time, you can toggle **Checked** on and off as part of a Click event procedure attached to a **Menu** control. You can also set the value of **Checked** in a startup procedure or in a form's Load event procedure.

For a **Menu** control, **Checked** is normally read/write at run time. But **Checked** is read-only for menu items that are exposed or supplied by Visual Basic to add-ins, such as the Add-In Manager command on the Add-Ins menu.

Count Property (VB Collections)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproCountC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCountX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCountS"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCountA"}

Returns the number of objects in a collection.

Syntax

object.**Count**

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

Remarks

You can use this property with a **For...Next** statement to carry out an operation on the forms or controls in a collection. For example, the following code moves all controls on a form 0.5 inches to the right (**ScaleMode** property setting is 1 or **vbTwips**):

```
For I = 0 To Form1.Controls.Count - 1
   Form1.Controls(I).Left = Form1.Controls(I).Left + 720
Next I
```

You can also use this kind of structure to quickly enable or disable all controls on a form.

When used with the **If TypeOf** statement, you can cycle through all controls and change, for example, the **Enabled** property setting of only the text boxes or the **BackColor** property setting of only the option buttons.

Enabled Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproEnabledC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproEnabledX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproEnabledA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproEnabledS"}

Returns or sets a value that determines whether a form or control can respond to user-generated events.

Syntax

object.**Enabled** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the form associated with the active form module is assumed to be *object.*
boolean	A Boolean expression that specifies whether *object* can respond to user-generated events.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) Allows *object* to respond to events.
False	Prevents *object* from responding to events.

Remarks

The **Enabled** property allows forms and controls to be enabled or disabled at run time. For example, you can disable objects that don't apply to the current state of the application. You can also disable a control used purely for display purposes, such as a text box that provides read-only information.

Disabling a **Timer** control by setting **Enabled** to **False** cancels the countdown set up by the control's **Interval** property.

For a **Menu** control, **Enabled** is normally read/write at run time. But **Enabled** is read-only for menu items that are exposed or supplied by Visual Basic to add-ins, such as the Add-In Manager command on the Add-Ins menu.

HelpContextID Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHelpContextIDC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHelpContextIDX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHelpContextIDA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHelpContextIDS"}

Returns or sets an associated context number for an object. Used to provide context-sensitive Help for your application.

Syntax

object.**HelpContextID** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the form associated with the active form module is assumed to be *object.*
number	A numeric expression that specifies the context number of the Help topic associated with *object*.

Settings

The settings for *number* are:

Setting	Description
0	(Default) No context number specified.
> 0	An integer specifying a valid context number.

Remarks

For context-sensitive Help on an object in your application, you must assign the same context number to both *object* and to the associated Help topic when you compile your Help file.

If you've created a Microsoft Windows operating environment Help file for your application and set the application's **HelpFile** property, when a user presses the F1 key, Visual Basic automatically calls Help and searches for the topic identified by the current context number.

The current context number is the value of **HelpContextID** for the object that has the focus. If **HelpContextID** is set to 0, then Visual Basic looks in the **HelpContextID** of the object's container, and then that object's container, and so on. If a nonzero current context number can't be found, the F1 key is ignored.

For a **Menu** control, **HelpContextID** is normally read/write at run time. But **HelpContextID** is read-only for menu items that are exposed or supplied by Visual Basic to add-ins, such as the Add-In Manager command on the Add-Ins menu.

Note Building a Help file requires the Microsoft Windows Help Compiler, which is included with the Visual Basic Professional Edition.

Index Property (Control Array)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproIndexC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproIndexX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproIndexA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproIndexS"}

Returns or sets the number that uniquely identifies a control in a control array. Available only if the control is part of a control array.

Syntax

object[(*number*)].**Index**

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression that evaluates to an integer that identifies an individual control within a control array.

Settings

The settings for *number* are:

Setting	Description
No value	(Default) Not part of a control array.
0 to 32,767	Part of an array. Specifies an integer greater than or equal to 0 that identifies a control within a control array. All controls in a control array have the same **Name** property. Visual Basic automatically assigns the next integer available within the control array.

Remarks

Because control array elements share the same **Name** property setting, you must use the **Index** property in code to specify a particular control in the array. **Index** must appear as an integer (or a numeric expression evaluating to an integer) in parentheses next to the control array name — for example, `MyButtons(3)`. You can also use the **Tag** property setting to distinguish one control from another within a control array.

When a control in the array recognizes that an event has occurred, Visual Basic calls the control array's event procedure and passes the applicable **Index** setting as an additional argument. This property is also used when you create controls dynamically at run time with the **Load** statement or remove them with the **Unload** statement.

Although Visual Basic assigns, by default, the next integer available as the value of **Index** for a new control in a control array, you can override this assigned value and skip integers. You can also set **Index** to an integer other than 0 for the first control in the array. If you reference an **Index** value in code that doesn't identify one of the controls in a control array, a Visual Basic run-time error occurs.

Note To remove a control from a control array, change the control's **Name** property setting, and delete the control's **Index** property setting.

Name Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproNameC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproNameX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproNameA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproNameS"}

Returns the name used in code to identify a form, control, or data access object. Read-only at run time

Syntax

object.**Name**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list. If *object* is omitted, the form associated with the active form module is assumed to be *object.*

Remarks

The default name for new objects is the kind of object plus a unique integer. For example, the first new **Form** object is Form1, a new **MDIForm** object is MDIForm1, and the third **TextBox** control you create on a form is Text3.

An object's **Name** property must start with a letter and can be a maximum of 40 characters. It can include numbers and underline (_) characters but can't include punctuation or spaces. Forms can't have the same name as another public object such as **Clipboard**, **Screen**, or **App**. Although the **Name** property setting can be a keyword, property name, or the name of another object, this can create conflicts in your code.

You can use a form's **Name** property with the **Dim** statement at run time to create other instances of the form. You can't have two forms with the same name at design time.

You can create an array of controls of the same type by setting the **Name** property to the same value. For example, when you set the name of all option buttons in a group to MyOpt, Visual Basic assigns unique values to the **Index** property of each control to distinguish it from others in the array. Two controls of different types can't share the same name.

Note Although Visual Basic often uses the **Name** property setting as the default value for the **Caption**, **LinkTopic**, and **Text** properties, changing one of these properties doesn't affect the others.

Parent Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproParentC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproParentX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproParentA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproParentS"}

Returns the form, object, or collection that contains a control or another object or collection.

Syntax

object.**Parent**

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

Remarks

Use the **Parent** property to access the properties, methods, or controls of an object's parent. For example:

```
MyButton.Parent.MousePointer = 4
```

The **Parent** property is useful in an application in which you pass objects as arguments. For example, you could pass a control variable to a general procedure in a module, and use the **Parent** property to access its parent form.

There is no relationship between the **Parent** property and the **MDIChild** property. There is, however, a parent-child relationship between an **MDIForm** object and any **Form** object that has its **MDIChild** property set to **True**.

Path Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPathC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPathX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPathA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPathS"}

Returns or sets the current path. Not available at design time. For the **App** object, read-only at run time.

Syntax

object.**Path** [= *pathname*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
pathname	A string expression that evaluates to the path name.

Remarks

The value of the **Path** property is a string indicating a path, such as C:\Ob or C:\Windows\System. For a **DirListBox** or **FileListBox** control, the default is the current path when the control is created at run time. For the **App** object, **Path** specifies the path of the project .VBP file when running the application from the development environment or the path of the .exe file when running the application as an executable file.

Use this property when building an application's file-browsing and manipulation capabilities. Setting the **Path** property has effects on a control similar to the MS-DOS **chdir** command — relative paths are allowed with or without a drive specification. Specifying only a drive with a colon (:) selects the current directory on that drive.

The **Path** property can also be set to a qualified network path without a drive connection using the following syntax:

*servername**sharename**path*

The preceding syntax changes the **Drive** property to a zero-length string ("").

Changing the value of **Path** has these effects:

- For a **DirListBox** control, generates a Change event.
- For a **FileListBox** control, generates a PathChange event.

Note For **DirListBox**, the return value of **Path** is different from that of `List(ListIndex)`, which returns only the selection.

Zoom Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproZoomC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproZoomX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproZoomA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproZoomS"}

Returns or sets the percentage by which printed output is to be scaled up or down. Not available at design time.

Syntax

object.**Zoom** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression that evaluates to the percentage by which printed output is to be scaled. The default is 0, which specifies that the printed page appears at its normal size.

Remarks

The **Zoom** property setting scales the size of the physical page up or down, by a factor of Zoom/100, to the apparent size of the printed output. For example, a letter-size page printed with **Zoom** set to 50 contains as much data as a page of the size 17 by 22 inches because the printed text and graphics are scaled to one-half their original height and width.

Note The effect of the properties of the **Printer** object depends on the driver supplied by the printer manufacturer. Some property settings may have no effect, or several different property settings may all have the same effect. Settings outside the accepted range may or may not produce an error. For more information, see the manufacturer's documentation for the specific driver.

Font Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFontC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFonts"}

Returns a **Font** object.

Syntax

object.**Font**

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

Remarks

Use the **Font** property of an object to identify a specific **Font** object whose properties you want to use. For example, the following code changes the **Bold** property setting of a **Font** object identified by the **Font** property of a **TextBox** object:

```
txtFirstName.Font.Bold = True
```

Container Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproStandardContainerC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproStandardContainerX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproStandardContainerA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproStandardContainerS"}

Returns or sets the container of a control on a **Form**. Not available at design time.

Syntax

Set *object*.**Container** [= *container*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
container	An object expression that evaluates to an object that can serve as a container for other controls, as described in Remarks.

Remarks

The following controls can contain other controls:

- **Frame** control
- **PictureBox** control.

Object Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproObjectExtdC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproObjectExtdX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproObjectExtdA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproObjectExtdS"}

Returns a reference to a property or method of a control which has the same name as a property or method automatically extended to the control by Visual Basic.

Syntax

object.**Object**[.*property* | .*method*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
property	Property of the control that is identical to the name of a Visual Basic-supplied property.
Method	Method of the control that is identical to the name of a Visual Basic-supplied method.

Remarks

Note The **Object** property returns the object which is the basis for the control without the properties or methods automatically extended to the control by Visual Basic. Therefore, you can also reference the controls' "custom" properties and methods through the **Object** property, such as `Print SSTab1.Object.Tabs`.

Visual Basic supplies some or all of a standard set of properties and methods to controls in a Visual Basic project. It is possible for a control or ActiveX component (such as Microsoft Excel or Microsoft Word) to define a property or method which has the same name as one of these standard properties or methods. When this occurs, Visual Basic automatically uses the property or method it supplies instead of the one with the same name defined in the control. The **Object** property allows you to bypass the Visual Basic-supplied property or method and use the identically named property or method defined in the control.

For example, the **Tag** property is a property supplied to all controls in a Visual Basic project. If a control in a project has the name `ctlDemo`, and you access the **Tag** property using this syntax:

```
ctlDemo.Tag
```

Visual Basic automatically uses the **Tag** property it supplies. However, if the control defines its own **Tag** property and you want to access that property, you use the **Object** property in this syntax:

```
ctlDemo.Object.Tag
```

Visual Basic automatically extends some or all of the following properties, methods, and events to controls in a Visual Basic project:

Properties

Align	**Height**	**Object**
Binding	**HelpContextID**	**Parent**
Bindings	**Index**	**TabIndex**
Cancel	**Left**	**TabStop**
Container	**LeftNoRun**	**TagParent**
DataChanged	**LinkItem**	**ToolTipText**
DataField	**LinkMode**	**Top**

DataSource	**LinkTimeout**	**TopNoRun**
Default	**LinkTopic**	**VisibleTabStop**
DragIcon	**Name**	**WhatsThisHelpID**
DragMode	**NegotiateLinkItem**	**Width**

Methods

Drag	**LinkSend**	**ShowWhatsThis**
LinkExecute	**Move**	**Zorder**
LinkPoke	**Refresh**	
LinkRequest	**SetFocus**	

Events

GotFocus	**LinkError**	**LinkOpen**
LinkClose	**LinkNotify**	**LostFocus**

If you use a property or method of a control and don't get the behavior you expect, check to see if the property or method has the same name as one of those shown in the preceding list. If the names match, check the documentation provided with the control to see if the behavior matches the Visual Basic-supplied property or method. If the behaviors aren't identical, you may need to use the **Object** property to access the feature of the control that you want.

ToolTipText Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproToolTipC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTooltipX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTooltipA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTooltips"}

Returns or sets a ToolTip.

Syntax

object.**ToolTipText** [= *string*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	A string associated with an object in the Applies To list. that appears in a small rectangle below the object when the user's cursor hovers over the object at run time for about one second.

Remarks

If you use only an image to label an object, you can use this property to explain each object with a few words.

At design time you can set the **ToolTipText** property string in the control's properties dialog box.

For the **Toolbar** and **TabStrip** controls, you must set the **ShowTips** property to True to display ToolTips.

ActiveForm Property Example

This example prints the time on the active child form in an **MDIForm** object. To try this example, create an **MDIForm**, draw a **PictureBox** control on it and a **CommandButton** control in the **PictureBox**. In Form1, set the **MDIChild** property to **True**. (You can also set **AutoRedraw** to **True** to keep text on the form even after covering it with another form.) Paste the appropriate code into the Declarations section of each form, and then press F5.

```
' Copy all code into the MDI form.
Private Sub MDIForm_Load ()
   Dim NewForm As New Form1  ' Create new instance of Form1.
   NewForm.Show
End Sub

Private Sub Command1_Click ()
   ' Print the time on the active form.
   ActiveForm.Print "The time is " & Format(Now, "Long Time")
End Sub
```

This example shows how you can use the **Clipboard** object in cut, copy, paste, and delete operations using buttons on a toolbar. To try this example, create a new project, then put **TextBox** and **CheckBox** controls on Form1, and then create a new MDI form. On the MDI form, place a **PictureBox** control, and then insert a **CommandButton** control in the **PictureBox**. Set the **Index** property of the **CommandButton** to 0 (creating a control array). Set the **MDIChild** property of Form1 to **True**.

To run the example, copy the code into the Declarations section of the **MDIForm**, and then press F5. Notice that when the **CheckBox** has the focus, the buttons don't work, since the **CheckBox** is now the active control instead of the **TextBox**.

```
Private Sub MDIForm_Load ()
   Dim I       ' Declare variable.
   Command1(0).Move 0, 0, 700, 300 ' Position button on toolbar.
   For I = 1 To 3 ' Create other buttons.
      Load Command1(I)  ' Create button.
      Command1(I).Move I * 700, 0, 700, 300   ' Place and size button.
      Command1(I).Visible = True   ' Display button.
   Next I
   Command1(0).Caption = "Cut"     ' Set button captions.
   Command1(1).Caption = "Copy"
   Command1(2).Caption = "Paste"
   Command1(3).Caption = "Del"
End Sub

Private Sub Command1_Click (Index As Integer)
   ' ActiveForm refers to the active form in the MDI form.
   If TypeOf ActiveForm.ActiveControl Is TextBox Then
      Select Case Index
         Case 0   ' Cut.
            ' Copy selected text to Clipboard.
            Clipboard.SetText ActiveForm.ActiveControl.SelText
            ' Delete selected text.
            ActiveForm.ActiveControl.SelText = ""
         Case 1   ' Copy.
            ' Copy selected text to Clipboard.
            Clipboard.SetText ActiveForm.ActiveControl.SelText
         Case 2   ' Paste.
            ' Put Clipboard text in text box.
```

```
            ActiveForm.ActiveControl.SelText = Clipboard.GetText()
         Case 3   ' Delete.
            ' Delete selected text.
            ActiveForm.ActiveControl.SelText = ""
      End Select
   End If
End Sub
```

Caption Property Example

This example changes the **Caption** property of a **CommandButton** control each time the user clicks the button. To try this example, paste the code into the Declarations section of a form containing a **CommandButton** named Command1, and then press F5 and click the button.

```
Private Sub Command1_Click ()
   ' Check caption, then change it.
   If Command1.Caption = "Clicked" Then
      Command1.Caption = "OK"
   Else
      Command1.Caption = "Clicked"
   End If
End Sub
```

Checked Property Example

This example displays and removes a check mark next to a menu item. To try this example, create a form with a **Menu** control that has one menu item (set both the **Caption** and **Name** properties to MyMenuItem), and then press F5 and choose the menu item.

```
Private Sub MyMenuItem_Click ()
   ' Turn check mark on menu item on and off.
   MyMenuItem.Checked = Not MyMenuItem.Checked
End Sub
```

Enabled Property Example

This example enables a **CommandButton** control whenever a **TextBox** control contains text. To try this example, paste the code into the Declarations section of a form with **CommandButton** and **TextBox** controls, and then press F5 and enter something into the text box.

```
Private Sub Form_Load ()
   Text1.Text = ""    ' Clear the text box.
   Command1.Caption = "Save" ' Put caption on button.
End Sub

Private Sub Text1_Change ()
   If Text1.Text = "" Then ' See if text box is empty.
      Command1.Enabled = False     ' Disable button.
   Else
      Command1.Enabled = True      ' Enable button.
   End If
End Sub
```

HelpContextID Property Example

This example uses topics in the Visual Basic Help file to demonstrate how to specify context numbers for Help topics. To try this example, paste the code into the Declarations section of a form that contains a **TextBox** control and a **Frame** control with an **OptionButton** control inside of it. Press F5. Once the program is running, move the focus to one of the controls, and press F1.

```
' Actual context numbers from the Visual Basic Help file.
Const winColorPalette = 21004' Define constants.
Const winToolbox = 21001
Const winCodeWindow = 21005

Private Sub Form_Load ()
   App.HelpFile = "VB.HLP"
   Frame1.HelpContextID = winColorPalette
   Text1.HelpContextID = winToolbox
   Form1.HelpContextID = winCodeWindow
End Sub
```

Index Property Example

This example starts with two **OptionButton** controls and adds a new **OptionButton** to the form each time you click a **CommandButton** control. When you click an **OptionButton**, the **FillStyle** property is set and a new circle is drawn. To try this example, paste the code into the Declarations section of a form that has two **OptionButton** controls, a **CommandButton**, and a large **PictureBox** control. Set the **Name** property of both **OptionButton** controls to **optButton** to create a control array.

```
Private Sub OptButton_Click (Index As Integer)
   Dim H, W ' Declare variables.
   Picture1.Cls   ' Clear picture.
   Picture1.FillStyle = Index      ' Set FillStyle.
   W = Picture1.ScaleWidth / 2     ' Get size of circle.
   H = Picture1.ScaleHeight / 2
   Picture1.Circle (W, H), W / 2   ' Draw circle.
End Sub

Private Sub Command1_Click ()
   Static MaxIdx  ' Largest index in array.
   If MaxIdx = 0 Then MaxIdx = 1   ' Preset MaxIdx.
   MaxIdx = MaxIdx + 1  ' Increment index.
   If MaxIdx > 7 Then Exit Sub     ' Put eight buttons on form.
   Load OptButton(MaxIdx)  ' Create new item in array.
   ' Set location of new option button under previous button.
   OptButton(MaxIdx).Top = OptButton(MaxIdx - 1).Top + 360
   OptButton(MaxIdx).Visible = True      ' Make new button visible.
End Sub
```

Parent Property Example

This example passes a control from a form that doesn't have the focus to a procedure in a module, and then displays the state of the control on the parent form. To try this example, create three forms: Form1, containing a **CommandButton** control, and Form2 and Form3, each containing a **CheckBox** control. You must also create a new module (click Add Module in the Project menu). Paste the code into the Declarations sections of the respective forms or module, and then press F5 to run the program.

```
' Enter this code into Form1.
Private Sub Form_Load ()
   Form2.Show  ' Display all forms.
   Form3.Show
   Form2.AutoRedraw = True
   Form3.AutoRedraw = True
End Sub

Private Sub Command1_Click ()
   ReadCheckBox Form2.Check1 ' Call procedure in other module
   ReadCheckBox Form3.Check1 ' and send control as argument.
End Sub

' Enter this code into Module1.
Sub ReadCheckBox (Source As Control)
   If Source.Value Then
      Source.Parent.Cls ' Clear parent form.
      Source.Parent.Print "CheckBox is ON."  ' Display on parent form.
   Else
      Source.Parent.Cls ' Clear parent form.
      Source.Parent.Print "CheckBox is OFF."  ' Display on parent form.
   End If
End Sub
```

Path Property Example

This example displays a list of files for the selected drive and directory. To try this example, paste the code into the Declarations section of a form that contains **DriveListBox**, **DirListBox**, and **FileListBox** controls. Press F5. Use the mouse to change the drive or directory.

```
Private Sub Drive1_Change ()
   Dir1.Path = Drive1.Drive  ' Set directory path.
End Sub

Private Sub Dir1_Change ()
   File1.Path = Dir1.Path  ' Set file path.
End Sub
```

Container Property Example

This example demonstrates moving a **CommandButton** control from container to container on a **Form** object. To try this example, paste the code into the Declarations section of a form that contains a **Frame** control, a **PictureBox** control and a **CommandButton**, and then press F5.

```
Private Sub Form_Click()
   Static intX As Integer
   Select Case intX
      Case 0
         Set Command1.Container = Picture1
         Command1.Top= 0
         Command1.Left= 0
      Case 1
         Set Command1.Container = Frame1
         Command1.Top= 0
         Command1.Left= 0
      Case 2
         Set Command1.Container = Form1
         Command1.Top= 0
         Command1.Left= 0
   End Select
   intX = intX + 1
End Sub
```

DisabledPicture Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDisabledPicturePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDisabledPicturePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDisabledPicturePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDisabledPicturePropertyS"}

Returns or sets a reference to a picture to display in a control when it is disabled. (That is, when its **Enabled** property is set to **False**.)

Syntax

object.**DisabledPicture** [= *picture*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
picture	A **Picture** object containing a graphic, as described in Settings.

Settings

The settings for *picture* are:

Setting	Description
(None)	(Default) No picture.
(Bitmap, icon, metafile)	Specifies a graphic. You can load the graphic from the Properties window at design time. At run time, you can also set this property by using the **LoadPicture** function on a bitmap, icon, or metafile, or by setting it to the **Picture** property of another control.

Remarks

The **DisabledPicture** property specifies a picture object to display when the control (such as a **CommandButton**) is disabled. The **DisabledPicture** property is ignored unless the **Style** property of the control is set to 1 (graphical).

The picture is centered horizontally and vertically on the control. If there is a caption as well as a picture, the picture is centered above the caption. If the picture object is too large to fit on the control, then it is clipped.

If no picture is assigned to the **DisabledPicture** property, but one is assigned to the **Picture** property, then a grayed version of that picture is displayed when the control is disabled.

DownPicture Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDownPicturePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDownPicturePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDownPicturePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDownPicturePropertyS"}

Returns or sets a reference to a picture to display in a control when it is clicked and in the down (depressed) position.

Syntax

object.**DownPicture** [= *picture*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
picture	A **Picture** object containing a graphic, as described in Settings.

Settings

The settings for *picture* are:

Setting	Description
(None)	(Default) No picture.
(Bitmap, icon, metafile)	Specifies a graphic. You can load the graphic from the Properties window at design time. At run time, you can also set this property by using the **LoadPicture** function on a bitmap, icon, or metafile, or by setting it to the **Picture** property of another control.

Remarks

The **DownPicture** property refers to a picture object that displays when the button is in the down state. The **DownPicture** property is ignored unless the **Style** property is set to 1 (graphical). Note that when an **OptionButton** or **CheckBox** control's **Style** property is set to graphical and its button depressed, the background of the button is dithered, but the picture on the button is not.

The picture is centered both horizontally and vertically on the button. If there is a caption included with the picture, the picture will be centered above the caption. If no picture is assigned to this property when the button is depressed, then the picture currently assigned to the **Picture** property is used. If no picture is assigned to either the **Picture** or **DownPicture** properties, then only the caption is displayed. If the picture object is too large to fit on the button, then it is clipped.

MaskColor Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMaskColorPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMaskColorPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMaskColorPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMaskColorPropertyS"}

Returns or sets a color in a button's picture to be a "mask" (that is, transparent).

Syntax

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

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
color	A value or constant 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
Normal RGB colors	Colors specified using the Color palette or by using the **RGB** or **QBColor** functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.
&H00C0C0C0	(Default) Light gray.

Remarks

This property is used only when the **UseMask** property is set to **True** and the button has a bitmap-style picture assigned to its **Picture** property. (Icons and metafiles already contain tranparency information.)

If the **MaskColor** property is changed at run time, the button will redraw itself with the new color acting as a mask.

UseMask Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproUseMaskPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproUseMaskPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproUseMaskPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproUseMaskPropertyS"}

Returns or sets a value that determines whether the color assigned in the **MaskColor** property is used as a "mask". (That is, used to create transparent regions.)

Syntax

object.**UseMask** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that specifies whether the color assigned to the **MaskColor** property is used as a mask.

Settings

The settings for *boolean* are:

Setting	Description
True	The color assigned to the **MaskColor** property is used as a mask, creating a transparent region wherever that color is.
False	(Default) The color assigned to the **MaskColor** property is ignored, and the color remains opaque.

CHAPTER 0

A Reference Template

*Insert existing text here and delete this text. **Do not remove** the following paragraph.

WhatsThisHelpID Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproWhatsThisHelpIDC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproWhatsThisHelpIDX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproWhatsThisHelpIDA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproWhatsThisHelpIDS"}

Returns or sets an associated context number for an object. Use to provide context-sensitive Help for your application using the What's This pop-up in Windows 95 Help.

Syntax

object.**WhatsThisHelpID** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression specifying a Help context number, as described in Settings.

Settings

The settings for *number* are:

Setting	Description
0	(Default) No context number specified.
>0	An integer specifying the valid context number for the What's This topic associated with the object.

Remarks

Windows 95 uses the What's This button in the upper-right corner of the window to start Windows Help and load a topic identified by the **WhatsThisHelpID** property.

WhatsThisButton Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproWhatsThisButtonC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproWhatsThisButtonX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproWhatsThisButtonA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproWhatsThisButtonS"}

Returns or sets a value that determines whether the What's This button appears in the title bar of a **Form** object. Read-only at run time.

Syntax

object.**WhatsThisButton**

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

Settings

The settings for the **WhatsThisButton** property are:

Setting	Description
True	Turns display of the What's This Help button on.
False	(Default) Turns display of the What's This Help button off.

Remarks

The **WhatsThisHelp** property must be **True** for the **WhatsThisButton** property to be **True**. In addition, the following properties must also be set as shown:

- **ControlBox** property = **True**
- **BorderStyle** property = Fixed Single or Sizable
- **MinButton** and **MaxButton** = **False**

– Or –

- **BorderStyle** property = Fixed Dialog

WhatsThisHelp Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproWhatsThisHelpC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproWhatsThisHelpX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproWhatsThisHelpA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproWhatsThisHelpS"}

Returns or sets a value that determines whether context-sensitive Help uses the What's This pop-up provided by Windows 95 Help or the main Help window. Read-only at run time.

Syntax

object.**WhatsThisHelp** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A value that determines if Help uses the What's This pop-up, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The application uses one of the What's This access techniques to start Windows Help and load a topic identified by the **WhatsThisHelpID** property.
False	(Default) The application uses the F1 key to start Windows Help and load the topic identified by the **HelpContextID** property.

Remarks

There are three access techniques for providing What's This Help in an application. The **WhatsThisHelp** property must be set to **True** for any of these techniques to work.

- Providing a What's This button in the title bar of the form using the **WhatsThisButton** property. The mouse pointer changes into the What's This state (arrow with question mark). The topic displayed is identified by the **WhatsThisHelpID** property of the control clicked by the user.
- Invoking the **WhatsThisMode** method of a form. This produces the same behavior as clicking the What's This button without using a button. For example, you can invoke this method from a command on a menu in the menu bar of your application.
- Invoking the **ShowWhatsThis** method for a particular control. The topic displayed is identified by the **WhatsThisHelpID** property of the control.

ShowWhatsThis Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthShowWhatsThisC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthShowWhatsThisX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthShowWhatsThisA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthShowWhatsThisS"}

Displays a selected topic in a Help file using the What's This popup provided by Windows 95 Help.

Syntax

object.**ShowWhatsThis**

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

Remarks

The **ShowWhatsThis** method is very useful for providing context-sensitive Help from a context menu in your application. The method displays the topic identified by the **WhatsThisHelpID** property of the object specified in the syntax.

ShowWhatsThis Method Example

This example displays the What's This Help topic for a **CommandButton** control by selecting a menu command from a context menu created for the button. Set the **WhatsThisHelp** property of the form to **True**. Place a **CommandButton** control on a form, create a menu using the Menu Editor with a top-level invisible item named mnuBtnContextMenu, and a sub-menu named mnuBtnWhatsThis with a caption of "What's This?".

```
Private ThisControl As Control

Private Sub Command1_MouseUp(Button As Integer, Shift As Integer, X As
Single, Y As Single)
      If Button = vbRightButton Then
         Set ThisControl = Command1
         PopupMenu mnuBtnContextMenu
      End If
   Set ThisControl = Nothing
End Sub

Private Sub mnuBtnWhatsThis_Click()
   ThisControl.ShowWhatsThis
End Sub
```

WhatsThisMode Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthWhatsThisModeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthWhatsThisModeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthWhatsThisModeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthWhatsThisModeS"}

Causes the mouse pointer to change into the What's This pointer and prepares the application to display What's This Help on the selected object.

Syntax

object.**WhatsThisMode**

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

Remarks

Executing the **WhatsThisMode** method places the application in the same state you get by clicking the What's This button in the title bar. The mouse pointer changes to the What's This pointer. When the user clicks an object, the **WhatsThisHelpID** property of the clicked object is used to invoke context-sensitive Help. This method is especially useful when invoking Help from a menu in the menu bar of your application.

WhatsThisMode Method Example

This example uses a command in a menu to change the mouse pointer to the What's This pointer and enable context-sensitive Help. To try the example, create a menu, and paste the code into the Click event of one of the **Menu** controls. Press F5, and click the menu command to toggle the application into the What's This state.

```
Private Sub mnuContextHelp_Click ()
   Form1.WhatsThisMode
End Sub
```

ShowInTaskbar Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproShowInTaskbarC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproShowInTaskbarX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproShowInTaskbarA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproShowInTaskbarS"}

Returns or sets a value that determines whether a **Form** object appears in the Windows 95 taskbar. Read-only at run time.

Syntax

object.**ShowInTaskbar**

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

Settings

The settings for the **ShowInTaskbar** property are:

Setting	Description
True	(Default) The **Form** object appears in the taskbar.
False	The **Form** object does not appear in the taskbar.

Remarks

Use the **ShowInTaskbar** property to keep dialog boxes in your application from appearing in the taskbar.

The default value for the **ShowInTaskbar** property assumes the default setting for the **BorderStyle** property of the **Form** object (Sizable). Changing the **BorderStyle** property may change the setting of the **ShowInTaskbar** property.

Accessibility for People with Disabilities

Microsoft is committed to making its products and services easier for everyone to use. This Help file provides information about the following features, products, and services, which make Microsoft Windows, Microsoft Windows NT, and Microsoft Visual Basic more accessible for people with disabilities:

Microsoft Visual Basic accessibility

Microsoft services for people who are deaf or hard-of-hearing.

Microsoft documentation in alternative formats.

Third-party utilities to enhance accessibility.

Customizing Windows or Windows NT.

Getting more information

Note The information in this section applies only to users who purchased Windows or Windows NT in the United States. If you purchased Windows or Windows NT outside the United States, your Windows package contains a subsidiary information card listing Microsoft Support Services telephone numbers and addresses. You can contact your subsidiary to find out whether the type of products and services described in this Help file are available in your area.

Microsoft Services for People Who Are Deaf or Hard-of-Hearing

If you are deaf or hard-of-hearing, complete access to Microsoft product and customer services is available through a text telephone (TT/TDD) service.

Sales Information

You can contact Microsoft Sales Information Center on a text telephone by dialing (800) 892-5234 between 6:30 A.M. and 5:30 P.M. Pacific time.

Technical Assistance

For technical assistance in the United States, you can contact Microsoft Support Network on a text telephone at (206) 635-4948 between 6:00 A.M. and 6:00 P.M. Pacific time, Monday through Friday, excluding holidays. In Canada, dial (905) 568-9641 between 8:00 A.M. and 8:00 P.M. Eastern time, Monday through Friday, excluding holidays. Microsoft support services are subject to Microsoft prices, terms, and conditions in place at the time the service is used.

Microsoft Documentation in Alternative Formats

In addition to the standard forms of documentation, many Microsoft products are also available in other formats to make them more accessible.

Many of the Visual Basic version 5 documents are also available in online viewer format, or in print. If you have difficulty reading or handling printed documentation, you can obtain many Microsoft publications from Recording for the Blind & Dyslexic, Inc. Recording for the Blind & Dyslexic distributes these documents to registered, eligible members of their distribution service, either on audio cassettes or on floppy disks. The Recording for the Blind & Dyslexic collection contains more than 80,000 titles, including Microsoft product documentation and books from Microsoft Press.

You can contact Recording for the Blind & Dyslexic at the following address or phone numbers for information about eligibility and availability of Microsoft product documentation and books from Microsoft Press:

Recording for the Blind & Dyslexic, Inc.

20 Roszel RoadPrinceton

NJ 08540

Phone:	(609) 452-0606
Fax:	(609) 987-8116
World Wide Web:	http://www.rfbd.org

Third-Party Utilities to Enhance Accessibility

A wide variety of third-party hardware and software products are available to make it easier to use personal computers. Among the different types of products available for the MS-DOS, Windows, and Windows NT operating systems are:

- Programs that enlarge or alter the color of information on the screen for people with visual impairments.
- Programs that describe information on the screen in Braille or synthesized speech for people who are blind or have difficulty reading.
- Hardware and software utilities that modify the behavior of the mouse and keyboard.
- Programs that enable people to "type" using a mouse or their voice.
- Word or phrase prediction software that enables people to type more quickly and with fewer keystrokes.
- Alternate input devices, such as single switch or puff-and-sip devices, for people who cannot use a mouse or a keyboard.

To learn more about these products, see the topic "Getting more information".

Customizing Windows or Windows NT

There are many ways you can customize Windows or Windows NT to make your computer more accessible.

- Beginning with Windows 95, accessibility features are built in to Windows. These features are useful for individuals who have difficulty typing or using a mouse, have moderately impaired vision, or who are deaf or hard-of-hearing. The features can be installed during setup, or you can add them later from your Windows 95 installation disks. Look up "accessibility" in the Windows Help Index for information about installing and using these features.
- Many of the features which make Windows 95 more accessible can be added to Windows NT, earlier versions of Microsoft Windows, and MS-DOS through Access Packs. You can download these files by modem, or you can order them on disks from Microsoft.
- You can also use Control Panel and other built-in features to adjust the appearance and behavior of Windows or Windows NT to suit varying vision and motor abilities. These include adjusting colors and sizes, sound volume, and the behavior of the mouse and keyboard.
- Dvorak keyboard layouts make the most frequently typed characters on a keyboard more accessible if you have difficulty using the standard "QWERTY" layout. There are three Dvorak layouts: one if you are a two-handed user, one if you type with your left hand only, and one if you type with your right hand only. You do not need to purchase any special equipment to use these features.

The specific features available, and whether they are built-in or must be obtained separately, depend on which operating system you are using.

For full documentation on the accessibility features available in the operating system you are using, obtain the appropriate application notes listed below. Accessibility features are also documented in the Microsoft Windows 95 Resource Kit and the Microsoft Windows NT Resource Kit.

Which Files to Download or Order

You can obtain these files by downloading them with your modem, or you can order them on disks by phone. Specific information about downloading or ordering these files immediately follows this list of files.

These files include:

- Application notes providing more complete documentation on ways to customize Windows and Windows NT
- Access Packs and Dvorak keyboard layouts provide additional features for versions of Windows or Windows NT in which they are not already included

For	You need
Application notes for Microsoft Wndows 95	Ww1062.exe
Application notes for Microsoft Windows NT 3.1 and 3.5 (includes Access Pack for Microsoft Windows NT)	Wn0789.exe
Application notes for Microsoft Windows for Workgroups 3.1	Wg0788.txt
Application notes for Microsoft Windows 3.1	Ww0787.txt
Application notes for Microsoft Windows 3.0	Ww0786.txt

Access Pack for Microsoft Windows 3.0 and 3.1	Accp.exe
Dvorak keyboard layouts for people who type with one hand (already included in Windows NT 3.5 and higher)	Ga0650.exe (Most network services)Ga0650.zip (Microsoft Download Service)

To Download the Access Packs, Application Notes, and Alternative Keyboard Layouts by Modem

If you have a modem, you can download these files from the following network services:

- Microsoft's World Wide Web site on the Internet. On the www.microsoft.com home page, click Support. click Knowledge Base, and select MS-DOS as the product. Enter KBFILE GA0650.EXE, and click GO! Open the article, and click the button to download the file.
- Microsoft's Internet servers, ftp.microsoft.com and gopher.microsoft.com, in /softlib/mslfiles
- MSN, The Microsoft Network online service
- CompuServe®, type GO MSL
- GEnie™
- Microsoft Download Service (MSDL), which you can reach by calling (206) 936-6735 any time except between 1:00 A.M. and 2:30 A.M. Pacific time.
- MSDL supports 1200, 2400, 9600, 14400, or 28800 baud rates (V.32 and V.42), with 8 data bits, no parity, and 1 stop bit.
- Various user-group bulletin boards (such as the bulletin-board services on the Association of PC User Groups network)

To Order the Access Packs, Application Notes, and Alternative Keyboard Layouts on Disks by Phone

If you do not have a modem, within the United States you can order the Access Packs, Application Notes, and Alternative Layouts on disks by calling Microsoft Sales Information Center at (800) 426-9400 (voice) or (800) 892-5234 (text telephone).

In Canada, you can call (905) 568-3503 or (905) 568-9641 (text telephone).

Getting More Information

Other products, services, and resources are available from Microsoft and other organizations.

Additional Microsoft Products and Services for People with Disabilities

For more information, contact:

Microsoft Sales Information Center One Microsoft Way Redmond, WA 98052-6393	World Wide Web: Voice telephone: Text telephone:	http://www.microsoft.com (800) 426-9400 (800) 892-5234

Directories of Computer Products that Help People with Disabilities

The Trace R&D Center at the University of Wisconsin–Madison produces a book and a compact disc that describe products that help people with disabilities use computers. The book, titled *Trace ResourceBook,* provides descriptions and photographs of about 2,000 products. The compact disc, titled *CO-NET CD*, provides a database of more than 18,000 products and other information for people with disabilities. It is issued twice a year.

To obtain these directories, contact:

Trace R&D Center University of Wisconsin S-151 Waisman Center 1500 Highland Avenue Madison, WI 53705-2280	World Wide Web: Fax:	http://trace.wisc.edu (608) 262-8848

Referrals to Assistive Technology Programs and Trained Evaluators

For general information and recommendations on how computers can help specific needs, you should consult a trained evaluator. An assistive technology program in your area will provide referrals to programs and services that are available to you.

To locate the assistive technology program nearest you, contact:

National Information System University of South Carolina Center for Developmental Disabilities Columbia, SC 29208	Voice/text telephone: Fax:	(803) 935-5231 (803) 935-5059

Microsoft Visual Basic Accessibility

Enlarging Toolbar Buttons

To view enlarged toolbar buttons, choose the Toolbars command from the View menu, select Customize, and then select the Large Icons check box on the Options tab.

Enlarging Text in the Code Window

To view enlarged text in the Code Window, choose the Options command from the Tools menu. Select the Editor Format tab, and then set the font and size you want.

Customizing the Keyboard

Microsoft Visual Basic for Windows supports Dvorak keyboard layouts, which make the most frequently typed characters more accessible.

Rearranging Windows

Windows in Visual Basic can be rearranged to best suit the way you work. You can select the windows you want docked by using the list on the Docking tab of the Options dialog, available on the Tools menu.

Customizing the Toolbox

You can customize the Toolbox by adding tabs to it or by adding controls to either the General tab or a custom tab, using the Components command from the Project menu.

Providing Visual Cues While Editing Code

Margin Indicators in the Code Window allow you to provide visual cues to certain actions while editing your code. The Indicators appear on the left margin of the Code window. You can turn the Margin Indicator Bar on and off in the Editor Format tab of the Options dialog box.

Adding Prompts for Parameters

You can also choose to display popups in the Code window that prompt you with the parameters to complete a function or statement. You can turn Parameter Info on and off using the Parameter Info command on the Edit menu.

Add Project Command (File Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdAddProjectCommandC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdAddProjectCommandS"}

Displays the Add Project dialog box so that you can add a new or existing project to the currently open project group. If you have only one project open, Visual Basic adds the project and creates a project group. A project group exists only if there is more than one project.

Note If you do not have a project group opened, Visual Basic automatically creates a project group containing the existing open project and the project you add.

A project group can contain any number of other projects. It is an easy way to load more than one project at a time

The first EXE project you add to your project group is automatically set as Startup unless you change it using the Set as Start Up command on the Project Explorer shortcut menu.

When you add an in-process project to the project group and reference it, the project appears in the References dialog box and is stored in the project file.

When you add a project to a group, Visual Basic checks if any of the other projects in the group have a reference to the binary version of the newly added group. If they do, the references automatically are changed to the source version.

Remove Project Command (Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdRemoveProjectC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdRemoveProjectS"}

Closes the selected project and removes it from the Project Explorer. Using this command also removes the project from the currently open project group.

<Project Name> Properties Command (Project Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdPropertiesCommandC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdPropertiesCommandS"}

Displays the Project Properties dialog box where you view the project properties for the selected project.

Toolbar shortcut:  .

File 1, 2, 3, 4 Command (File Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdFile1234C;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdFile1234S"}

Lists the four most recently used projects (.vbp) or project groups (.vbg).

Add Module and Add Class Module Commands (Project Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdAddModuleProjectC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdAddModuleProjectS"}

Displays the Add Module or Add Class Module dialog box so you can insert a new or existing module or class module into your active project.

Toolbar shortcuts: ⊞ (Add Module command) and

⊞ (Add Class Module command).

Add ActiveX Designer Command (Project Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdAddOLEDesignerProjectC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdAddOLEDesignerProjectS"}

Displays a list of available ActiveX Designers from which to choose the one you want to add to your project.

Note You must add an ActiveX designer using the Components command on the Project menu before this command is available.

Add Property Page Command (Project Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdAddPropertyPageProjectC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdAddPropertyPageProjectS"}

Displays the Add Property Page dialog box so you can insert new or existing property pages into your active project. Use the Property Page Wizard to build new property pages.

Toolbar shortcut:  .

Components Command (Project Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdOLEComponentsProjectC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdOLEComponentsProjectS"}

Displays the Components dialog box from which you can add controls, designers, or insertable objects (such as a Microsoft Word Document) to the Toolbox. You can also reference loaded control projects.

Remove <Item> Command (Project Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"IDD_CheckInC;IDD_CheckOutC;vbcmdRemoveItemProjectC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdRemoveItemProjectS"}

Removes the currently selected item from your project. If you made changes to it since last saving it, you are asked whether you want to save the changes first.

The file remains in all the other projects to which it has been added. Files that have been removed from a project remain stored on your hard disk until you delete them, following the standard procedures of your operating system.

When removing a module or resource file from a project, make sure the remaining code doesn't refer to the removed item.

Note Use the Components dialog box to remove an ActiveX control from the Toolbox. If you still have an ActiveX control on a form, you can't remove the control from the Toolbox.

Available only at design time.

Note If you are using an integrated source code control program, such as Microsoft SourceSafe (available with the Enterprise Edition), when you remove a file from your Visual Basic project, a dialog may prompt you to remove the file from the version control project.

With Microsoft SourceSafe, if the .vbp file is checked out, the references to the removed file(s) are automatically deleted. If the .vbp file isn't checked out, when you remove a file, the .vbp file will still contain a reference to the deleted file, causing an error the next time you load your project.

Set as Start Up Command (Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdSetAsStartUpShortcutC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdSetAsStartUpShortcutS"}

Makes the selected project the project to run when the user presses F5 or the Start button on the Standard or Debug toolbar. The start up project appears in bold text in the Project Explorer.

If you do not identify a project as the start up project, the first EXE project that is added to the group is identified, by default, as the start up project. Adding additional EXE project to the project group does not change the start up project.

Form Layout Window Command (View Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdFormLayoutWindowViewC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdFormLayoutWindowViewS"}

Displays the Form Layout window where you can preview and position your form as it will appear in your application. Use the Resolution Guide command on the shortcut menu to preview it using different resolutions.

Note You can view only those resolutions lower than the resolution at which your monitor is set.

Toolbar shortcut: .

Paste Link Command (File Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdPasteLinkCommandFileMenuC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdPasteLinkCommandFileMenuS"}

Supports pasting a link to a valid DDE source. Active only when the Clipboard contains a valid DDE source, and the selected control is a valid DDE link.

You can also use Paste Link to link data from another application using the OLE container control. Once you copy valid data onto the Clipboard (a paragraph from a word processor or a range of cells from a spreadsheet), you can select the OLE container control on your form, and then choose Paste Link to link the data.

Available only at design time.

Procedure Attributes Command (Tools Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdProcedureAttributesToolsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdProcedureAttributesToolsS"}

Opens the Procedure Attributes dialog box where you can set the attributes for each property and method specified for an item.

You can also use this command when you want to set the **Value** property for classes.

Note You must use this command to set the **Value** property as the default if you want the control host to find it.

Insert File Command (Edit Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdInsertFileEditC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdInsertFileEditS"}

Opens the Insert File dialog box so that you can insert text from an existing file at the current pointer position in the Code window.

Not available at run time or when no editor is open.

Print Setup Command (File Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdPrintSetupFileC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdPrintSetupFileS"}

Displays the standard Print Setup dialog box with options to specify the printer, page orientation, paper size, and paper source, as well as other printing options.

Available only at design time.

Toolbar shortcut:  .

Add Tab, Rename Tab, and Delete Tab Commands (Shortcut Menu)

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

Allows you to add, rename, or modify the tabs in the Toolbox. You cannot rename or delete the General tab.

Add Tab Opens a dialog box where you can add and name a new tab. The new tab appears below the General tab.

Rename Tab Opens a dialog box where you can type a new name for the selected tab.

Delete Tab Deletes the selected tab.

Move Up and Move Down Commands (Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdMoveUpMoveDownShortcutC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdMoveUpMoveDownShortcutS"}

Moves the selected tab of the Toolbox up or down one position each time you click it.

Update User Controls Command (Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdUpdateUserControlsShortcutC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdUpdateUserControlsShortcutS"}

Updates the user controls on your form. Use this command to make sure that the changes you make to your user controls are reflected on the forms using them.

Using the Update User Controls command closes and re-opens the form.

Add Command (Shortcut Menu)

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

Displays a submenu where you can choose which of the following items you want to add to your project.

- Form
- MDI Form
- Module
- Class Module
- User Control
- Property Page
- User Document
- ActiveX Designer
- File

Align To Grid Command (Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdAlignToGridShortcutC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdAlignToGridShortcutS"}

Aligns the top left of the selected objects on your form to the closest grid. The object is not resized.

You can change the grid settings from the General tab of the Options dialog box.

Toolbar shortcut: ⊞

Default Command (Color Palette Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdDefaultColorPaletteShortcutC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdDefaultColorPaletteShortcutS"}

Changes the colors to those specified in the Control Panel.

Resolution Guide (Form Layout Window Shortcut Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdResolutionGuideFLWShortcutC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdResolutionGuideFLWShortcutS"}

Displays dotted lines showing the right and lower borders of standard resolutions based on the resolution of your monitor. You can size and position your forms so that they fit appropriately with the resolution for which you are creating your application.

Note If your monitor is set for 640x480, you will not see the guidelines. The guidelines only appear with resolutions lower than those of your monitor.

Toolbars Command (View Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdToolbarsViewC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdToolbarsViewS"}

Displays a submenu with a list of the toolbars – Debug, Edit, Form Editor, Standard, any custom toolbars and the Customize command. You can choose a toolbar or use the Customize command to modify or create your own toolbar or menu.

Add Procedure Command (Tools Menu)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcmdAddProcedureToolsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcmdAddProcedureToolsS"}

Inserts a new **Sub**, **Function**, **Property**, or Event procedure into the active module.

Not available if the module is not active.

Toolbar shortcut:  .

Properties Command (Shortcut Menu)

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

Displays the Properties window.

If you select your form in the Object Browser, the Properties command displays the Member Options dialog box.

Description Command (Shortcut Menu)

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

Turns the description pane of the Properties window off and on.