This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Data Property

See Also Example Applies To

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

Data Property (MSChart)

See Also Example Applies To

Returns or sets a value that is inserted into the current data point in the data grid of a chart.

Syntax

object.**Data** [= *value*]

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

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
value	Integer. The data point value.

Remarks

If the current data point already contains a value, it is replaced by the new value. The chart is redrawn to reflect the new value for the current data point.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Database Property

See Also Example Applies To

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

© 2017 Microsoft

Visual Basic Reference

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
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DatabaseName Property

See Also Example Applies To

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

© 2017 Microsoft

Visual Basic Reference

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
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DataBindingBehavior Property

See Also Example Applies To

Sets a value that determines if an object can be bound to a data source. Only available at design time.

Syntax

object.**DataBindingBehavior** [= *number*]

The **DataBindingBehavior** 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 data binding behavior, as described in Settings.

Settings

The settings for *number* are:

Constant	Setting	Description
vbNone	0	(Default) The object can't be bound to a data source.
vbSimpleBound	1	The object can be bound to a data source using simple binding.
vbComplexBound	2	The object can be bound to a data source using complex binding.

Remarks

Use the **DataBindingBehavior** property when you want an object to act as a consumer of data provided by objects. A data consumer may be either simple bound (binding to single fields) or complex bound (binding to a rowset).

When **DataBindingBehavior** is set to 1 (**vbSimpleBound**), the **PropertyChanged** event and the **CanPropertyChange** method are added to the object's procedures.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DataBindings Property

See Also Example Applies To

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.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DataField Property

See Also Example Applies To

Returns or sets the name of a field that a data consumer will be bound to.

Syntax

object.**DataField** [= *string*]

The **DataField** 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 name of one of the fields in the **Recordset** object specified by a data source.

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 source, and the **DataField** property specifies a valid field name in the **Recordset** object created by the data source. 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:

```
Adodc1.RecordSource = "Select AVG(Sales)  " _
   & " AS AverageSales From SalesTable"
Text1.DataField = "AverageSales"
Adodc1.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.

This documentation is archived and is not being maintained.

Visual Basic: DataGrid Control

Visual Studio 6.0

DataField Property (DataGrid Control, Column Object)

See Also Example Applies To

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:

```
Adodc1.RecordSource = "Select AVG(Sales)  " _
   & " AS AverageSales From SalesTable"
Text1.DataField = "AverageSales"
Adodc1.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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

DataField Property (MSHFlexGrid)

SeeAlso Example Applies To

Returns the name of the field bound to the specified column in the specified band. This property is read-only.

Syntax

object.**DataField(***number, index***)** [= *string*]

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

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
number	A Long value that specifies the band that contains the column to affect.
index	A Long value that either specifies the column to get, or sets the binding field.
string	A string expression that evaluates the name of one of the fields in the Recordset object specified by the data provider.

Remarks

When *number* is unspecified, it defaults to 0. Hence, when the **MSHFlexGrid** is not bound to a hierarchical Recordset, using 0 and not specifying *number* both have the same result.

If the **MSHFlexGrid** is not bound and the *number* and *index* are valid, **DataField** returns an empty string.

If a **Refresh** is executed after the **DataProvider** property is changed, the Recordset object may have different fields. This may invalidate the **DataField** settings of the bound columns and produce a trappable error.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataRepeater Control

Visual Studio 6.0

DataField Property (RepeaterBinding Object)

See Also Example Applies To

Returns or sets a **DataField** to be bound to a property.

Syntax

object.**DataField** [=*string*]

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

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
string	The name of the property to bind to.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataRepeater Control

Visual Studio 6.0

DataFields Property

See Also Example Applies To

Returns an array of **DataField** names of the **DataSource**.

Syntax

object.**DataFields**(*index*)

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	The index of an element in the array. The array is 0-based.

Return Type

String

Remarks

The **DataFields** array contains the names of all bindable properties of the repeated control (see **RepeatedControlName**).

Use the **Ubound** method to determine how many elements are in the array. Since the array is 0-based, use the following code:

```
Debug.Print Ubound(DataRepeater1.DataFields) + 1
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DataFormat Property

See Also Example Applies To

Sets or returns the **StdDataFormat** object to which a bound object is attached. Read/write both at design time and run time.

Syntax

object.**DataFormat** = *formatobject*

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

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
formatobject	Required. A **StdDataFormat** object.

Remarks

The **DataFormat** property can also be set through a property page from the Properties window. If the setting in code differs from the property page, the property page setting is used for the first record fetched, and from then on the values in code will be used.

The **DataFormat** property of the **DataCombo** control is an **Extender** property. Therefore it's always visible on the property sheet and can be set in code. However the **DataCombo** control only formats the top item in its list. This can be disconcerting to the end user who sees a formatted top item, but is only given a list of unformatted items to select from. The formatted item may also mislead end users who assume the item will be entered in the database as formatted. For these reasons, it's advisable to not set the **DataFormat** property when using the **DataCombo** control.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataRepeater Control

Visual Studio 6.0

DataFormat Property (RepeaterBinding Object)

See Also Example Applies To

Returns or sets a reference to a **DataFormat** object.

Syntax

object.**DataFormat** [=*dataformat*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
dataformat	Optional. A reference to a **DataFormat** object.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DataFormats Property

See Also Example Applies To

Sets or returns the **StdDataFormat** objects to which a bound object is attached. Read/write both at design time and run time.

Syntax

object.**DataFormats**

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

Remarks

Complex bound objects can be bound to multiple **StdDataFormat** objects. The **DataGrid**, for example, has one binding per column. The **DataFormats** property allows access to multiple **StdDataFormat** objects bound to a single control.

The **DataFormats** property can also be set through a property page from the Properties window. If the setting in code differs from the the property page, the property page setting is used for the first record fetched, and from then on the values in code will be used.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

DataGrid Property

See Also Example Applies To

Returns a reference to a **DataGrid** object that describes the data grid associated with a chart.

Syntax

object.**DataGrid**

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DataChanged Property

See Also Example Applies To

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)

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DataChanged Property (ActiveX Controls)

See Also Example Applies To

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** [= *boolean*]

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 *boolean* 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

Boolean

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataGrid Control

Visual Studio 6.0

DataChanged Property (DataGrid)

See Also Example Applies To

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.

Data Type

Boolean

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DataMember Property

See Also Example Applies To

Returns or sets a specified data member from among several offered by the data provider.

Syntax

object.**DataMember** [= *string*]

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
string	The name of a data member.

Remarks

A data provider can have multiple sets of data that that a data consumer can choose to bind to. Each set of data is called a "data member," and is identified by a unique string.

For example, when using a **Data Environment** that contains several **Command** objects as a **DataSource**, the **DataMember** specifies which **Command** object to use.

When using a class module or a user control as a data source, program the **GetDataMember** to return an appropriate data member. The event's *DataMember* argument contains the value of the **DataMember** property. By querying the argument, you can determine what data member is requested, and pass back the appropriate data through the *Data* argument.

© 2017 Microsoft

Visual Basic Reference

BindingCollection Object, DataMembers Collection Example

The example uses a class module as a data source. When code to set the **DataSource** and **DataMember** properties of two **Binding** objects executes, the class module's Initialize event occurs; two ADO recordsets are created in that event, and the names of the recordsets are added to the **DataMembers** collection. The GetDataMember event and its arguments are used to return data to the data consumer.

To try the example, on the **Project** menu, click **References**, and set a reference to **Microsoft Data Binding Collection** and **Microsoft ActiveX Data Objects**. On the Project menu, click **Add Class Module**. Change the name of the class to MyDataClass, and set the **DataSourceBehavior** property to **vbDataSource**. Then draw two **TextBox** controls on a form. Paste the code into the **Form** object's code module.

```
Option Explicit
' Declare the object variables, one for a Class module named MyDataClass,
' and two more for each BindingCollection object one for each
' recordset).

Private clsData As New MyDataClass               ' Class module
Private bndColProducts As New BindingCollection  ' Bindings Collection
Private bndColSuppliers As New BindingCollection ' Bindings Collection

Private Sub Form_Load()
   ' Set DataSource and DataMember properties for each Bindings
   ' collection object.
   With bndColProducts
      .DataMember = "Products"
      Set .DataSource = clsData
      .Add Text1, "Text", "ProductName" ' Bind to a TextBox.
   End With

   With bndColSuppliers
      .DataMember = "Suppliers"
      Set .DataSource = clsData
      .Add Text2, "Text", "CompanyName" ' Bind to a TextBox.
   End With

   ' Change the Caption of Command1
   Command1.Caption = "MoveNext"

End Sub

Private Sub Command1_Click()
   clsData.MoveNext
End Sub
```

Paste the code below into the MyDataClass module. The **DataSourceBehavior** property must be set to **vbDataSource** in order to see the GetDataMember event. Run the project.

```
Option Explicit
' Declare object variables for ADO Recordset and Connection objects.
```

```
Private WithEvents rsProducts As ADODB.Recordset
Private WithEvents rsSuppliers As ADODB.Recordset
Private cnNwind As ADODB.Connection

Private Sub Class_Initialize()
   ' Add strings to the DataMembers collection.
   With DataMembers
      .Add "Products"
      .Add "Suppliers"
   End With

   ' Set Recordset objects.
   Set rsProducts = New ADODB.Recordset
   Set rsSuppliers = New ADODB.Recordset
   Set cnNwind = New ADODB.Connection

   ' Set the Connection object parameters.
   With cnNwind
      ' The Nwind.mdb that comes with Visual Basic must be installed on
      ' the computer or the code will fail. Otherwise alter the path to
      ' find the file on the computer.
      .Provider = "Microsoft.Jet.OLEDB.3.51"
      .Open "C:\Program Files\DevStudio\VB\Nwind.mdb"
   End With

   ' Open the recordset objects.
   rsSuppliers.Open "SELECT * FROM Suppliers", cnNwind, _
   adOpenStatic, adLockOptimistic
   rsProducts.Open "SELECT * FROM Products", cnNwind, _
   adOpenStatic, adLockOptimistic

End Sub

' The GetDataMember occurs when the DataSource property of a data
' consumer is set. In this case, the Bindings collection object is
' the consumer.
Private Sub Class_GetDataMember(DataMember As String, Data As Object)
   Select Case DataMember
   Case "Products"
      Set Data = rsProducts
   Case "Suppliers"
      Set Data = rsSuppliers
   Case ""
      ' Provide a default record source when no Data Member is specified.
      Set Data = rsProducts
   End Select

End Sub

Public Function MoveNext()
   If rsProducts.EOF Then
      rsProducts.MoveFirst
   Else
      rsProducts.MoveNext
   End If
End Function

Private Sub rsProducts_MoveComplete(ByVal adReason As _
ADODB.EventReasonEnum, ByVal pError As ADODB.Error, adStatus As _
ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)
   ' Keep the two recordsets in sync. The first textbox displays
   ' the supplier of the product. If the SupplierID for both
   ' recordsets are equivalent, no change needed. Otherwise,
   ' move to first record and test for SupplierID. This example
   ' is for demonstration only as the method is not the most
   ' efficient.

   If rsSuppliers("SupplierID").Value = _
   pRecordset("SupplierID").Value Then Exit Sub

   rsSuppliers.MoveFirst
   Do While Not rsSuppliers.EOF
      If rsSuppliers("SupplierID").Value = _
      pRecordset("SupplierID").Value Then
         Exit Sub
      Else
         rsSuppliers.MoveNext
      End If
   Loop
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DataMember Property (ActiveX Controls)

See Also Example Applies To

Returns or sets a specified data member from among several offered by the data provider.

Syntax

object.**DataMember** [= *string*]

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
string	The name of a data member.

Remarks

A data provider can have multiple sets of data that a data consumer can choose to bind to. Each set of data is called a "data member," and is identified by a unique string.

For example, when using a **Data Environment** that contains several **Command** objects as a **DataSource**, the **DataMember** specifies which **Command** object to use.

When using a class module or a user control as a data source, program the **GetDataMember** to return an appropriate data member. The event's *DataMember* argument contains the value of the **DataMember** property. By querying the argument, you can determine what data member is requested, and pass back the appropriate data through the *Data* argument.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DataMembers Property

See Also Example Applies To

Returns a reference to the **DataMembers** Collection.

Syntax

object.**DataMembers**

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

Remarks

A data provider can have multiple sets of data that a data consumer can choose to bind to. Each set of data is called a "data member," and is identified by a unique string.

The **DataMembers** collection contains the names of all data members accessible to the data consumer.

© 2017 Microsoft

Visual Basic Reference

DataMembers Property, Add Method Example

See Also

The example adds two data members to a **DataMembers** collection using the **Add** method.

```
Option Explicit
' Declare object variables for ADO recordsets, and a Connection object.
Private WithEvents rsProducts As ADODB.Recordset
Private WithEvents rsSuppliers As ADODB.Recordset
Private cnNwind As ADODB.Connection

Private Sub Class_Initialize()
   ' Add DataMember names to the DataMembers collection.
   With DataMembers
      .Add "Products"
      .Add "Suppliers"
   End With

   ' Set Recordset objects.
   Set rsProducts = New ADODB.Recordset
   Set rsSuppliers = New ADODB.Recordset

   ' Code to open recordsets not shown.

End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

DataPointLabel Property

See Also Example Applies To

Returns a reference to a **DataPointLabel** object that describes a label on an individual chart data point.

Syntax

object.**DataPointLabel**

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

DataPoints Property

See Also Example Applies To

Returns a reference to a **DataPoint** Collection that describes the data points within a chart series.

Syntax

object.**DataPoints**

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

DataSeriesInRow Property

See Also Example Applies To

Returns or sets a value that indicates whether series data is being read from a row or a column in a data grid associated with a chart.

Syntax

object.**DataSeriesInRow** [= *boolean*]

The **DataSeriesInRow** 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 controls how series data is read, as described in Settings.

Settings

The settings for *boolean* are:

Setting	**Description**
True	Series data is being read from a row in a data grid.
False	(Default) Series data is being read from a column.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DataSource Property

See Also Example Applies To

Returns or sets a data source through which a data consumer is bound to a database.

Syntax

object.**DataSource** [=*datasource*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
datasource	An object reference that qualifies as a data source, including **ADO Recordset** objects, and classes or user controls defined as data sources (**DataSourceBehavior** property = **vbDataSource**).

Remarks

Use the **Set** statement to set the **DataSource** property, as shown below:

```
Set Text1.DataSource = ADODC1
```

Note Two older controls, the **Data** control and **RemoteData** control, can be used as data sources, however you cannot set the **DataSource** property of another control or object to either of these controls at run time. For example, the following code will fail:

```
Set Text1.DataSource = Data1    ' Will fail! You can't set DataSource at
                               ' run time to an intrinsic Data control.
```

To use either the **Data** control or **RemoteData** control as a data source, you can set the **DataSource** property of bound controls at design time only.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DataSource Property (ActiveX Controls)

See Also Example Applies To

Returns or sets a data source through which a data consumer is bound to a database.

Syntax

object.**DataSource** [=*datasource*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
datasource	An object reference that qualifies as a data source, including **ADO Recordset** objects, and classes or user controls defined as data sources (**DataSourceBehavior** property = **vbDataSource**).

Remarks

Use the **Set** statement to set the **DataSource** property, as shown below:

```
Set DataGrid1.DataSource = ADODC1
```

Note Two older controls, the **Data** control and **RemoteData** control, can be used as data sources, however you cannot set the **DataSource** property of another control or object to either of these controls at run time. For example, the following code will fail:

```
Set Text1.DataSource = Data1    ' Will fail! You can't set DataSource at
                               ' run time to an intrinsic Data control.
```

To use either the **Data** control or **RemoteData** control as a data source, you can set the **DataSource** property of bound controls at design time only.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DataSourceBehavior Property

See Also Example Applies To

Sets a value that determines if an object can act as a source of data for other objects. May only be set at design time.

Syntax

object.**DataSourceBehavior** [= *number*]

The **DataSourceBehavior** 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 data source behavior, as described in Settings.

Settings

The settings for *number* are:

Constant	Setting	Description
vbNone	0	(Default) The object can't act as a data source.
vbDataSource	1	The object can act as a data source.

Remarks

Use the **DataSourceBehavior** property when you want an object to act as a source of data for other objects. When **DataSourceBehavior** is set to 1 (**vbDataSource**), the GetDataMember event is added to the objects procedures.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: RDO Data Control

Visual Studio 6.0

DataSourceName Property (Remote Data)

See Also Example Applies To

Returns or sets the data source name for a **RemoteData** control.

Syntax

object.**DataSourceName** [= *datasourcename*]

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

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
datasourcename	A string expression that indicates a registered data source name.

Remarks

This property can be left blank if the **RemoteData** control's **Connect** property identifies a data source name (DSN) registered in the Windows Registry (32-bit) or if you create a DSN-less connection that provides all required information in the **Connect** property.

Once the **rdoConnection** is opened by the **RemoteData** control, the **DataSourceName** property contains the DSN used to establish the connection it may be different from the value set before the connection is opened, because a user might select a data source from a list of valid DSN entries during the connection process.

If you change this property after the control's **rdoConnection** object is open, you must use the **RemoteData** control's **Refresh** method to open a new connection to the data source.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DataText Property

See Also Example Applies To

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.

© 2017 Microsoft

Visual Basic Reference

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
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic for Applications Reference

Visual Studio 6.0

DateCreated Property

See Also Example Applies To Specifics

Description

Returns the date and time that the specified file or folder was created. Read-only.

Syntax

object.**DateCreated**

The *object* is always a **File** or **Folder** object.

Remarks

The following code illustrates the use of the **DateCreated** property with a file:

```
Sub ShowFileInfo(filespec)
    Dim fs, f, s
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFile(filespec)
    s = "Created: " & f.DateCreated
    MsgBox s
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic for Applications Reference

Visual Studio 6.0

DateLastAccessed Property

See Also Example Applies To Specifics

Description

Returns the date and time that the specified file or folder was last accessed. Read-only.

Syntax

object.**DateLastAccessed**

The *object* is always a **File** or **Folder** object.

Remarks

The following code illustrates the use of the **DateLastAccessed** property with a file:

```
Sub ShowFileAccessInfo(filespec)
    Dim fs, f, s
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFile(filespec)
    s = UCase(filespec) & vbCrLf
    s = s & "Created: " & f.DateCreated & vbCrLf
    s = s & "Last Accessed: " & f.DateLastAccessed & vbCrLf
    s = s & "Last Modified: " & f.DateLastModified
    MsgBox s, 0, "File Access Info"
End Sub
```

Important This method depends on the underlying operating system for its behavior. If the operating system does not support providing time information, none will be returned.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic for Applications Reference

Visual Studio 6.0

DateLastModified Property

See Also Example Applies To Specifics

Description

Returns the date and time that the specified file or folder was last modified. Read-only.

Syntax

object.**DateLastModified**

The *object* is always a **File** or **Folder** object.

Remarks

The following code illustrates the use of the **DateLastModified** property with a file:

```
Sub ShowFileAccessInfo(filespec)
    Dim fs, f, s
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFile(filespec)
    s = UCase(filespec) & vbCrLf
    s = s & "Created: " & f.DateCreated & vbCrLf
    s = s & "Last Accessed: " & f.DateLastAccessed & vbCrLf
    s = s & "Last Modified: " & f.DateLastModified
    MsgBox s, 0, "File Access Info"
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

Day Property

See Also Example Applies To

Returns or sets a value that specifies the current day number.

Syntax

object.**Day** [= *number*]

The **Day** 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 indicating the day number.

Remarks

The **Day** property can be set to any integer from 1 to 31.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

DayBold Property

See Also Example Applies To

Returns or sets a value that determines if a displayed day is bold.

Syntax

object.**DayBold**(date) [= *boolean*]

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

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
date	A date expression specifying a date found in the **VisibleDays** property.
boolean	A Boolean expression specifying whether or not the date is bolded.

Settings

The settings for *boolean* are:

Setting	**Description**
True	The date is displayed in bold.
False	(Default) The date is not displayed in bold.

Remarks

The **DayBold** property is an array that corresponds to the **VisibleDays** property. Each Boolean element indicates whether its corresponding date should be displayed in bold.

Only dates that are currently displayed are valid. Valid dates can be found by looking in the **VisibleDays** property.

As you move from month to month, the information in this property is not preserved.

© 2017 Microsoft

Visual Basic: Windows Controls

DayBold Property Example

The following code sets the first and last displayed days to bold. To try the example, place a MonthView control on a form, paste the code into the Declarations section. Run the project and double-click the form.

```
Private Sub Form_DblClick()
   With MonthView1
      .DayBold(MonthView1.VisibleDays(1)) = True
      .DayBold(MonthView1.VisibleDays(42)) = True
   End With
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

DayOfWeek Property

See Also Example Applies To

Returns or sets a value that specifies the current day of week.

Syntax

object.**DayOfWeek** [= *number*]

The **DayOfWeek** 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 day of the week, as shown in Settings below.

Settings

The settings for *number* are:

Constant	Value	Description
mvwSunday	1	(Default) Sunday
mvwMonday	2	Monday
mvwTuesday	3	Tuesday
mvwWednesday	4	Wednesday
mvwThursday	5	Thursday
mvwFriday	6	Friday
mvwSaturday	7	Saturday

Remarks

The **DayOfWeek** property can be set to any integer from 1 to 7.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DEAggregates Property

See Also Example Applies To

Returns a reference to the DEAggregates collection. Read-only.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DECommands Property

See Also Example Applies To

Returns a reference to the DECommands collection. Read-only.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DEConnections Property

See Also Example Applies To

Returns a reference to the DEConnections collection. Read-only.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Default Property

See Also Example Applies To

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.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DefaultBind Property

See Also Example Applies To

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.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DefaultCancel Property

See Also Example Applies To

Returns or sets a value determining if a control can act as a standard command button. The **DefaultCancel** property is read/write at the controls authoring time, and not available at the controls 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	(Default) The control cannot act as a default or cancel command button. No constituent control can have its **Default** or **Cancel** property set to **True**.

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 controls 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 controls 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 controls AmbientChanged event procedure to detect changes in the **DisplayAsDefault** property, and the controls appearance adjusted accordingly.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DefaultCursorType Property (Data Control)

See Also Example Applies To

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

This documentation is archived and is not being maintained.

Visual Basic: CommonDialog Control

Visual Studio 6.0

DefaultExt Property

See Also Example Applies To

Returns or sets the default filename extension for the dialog box.

Syntax

object.**DefaultExt** [= *string*]

The **DefaultExt** 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 file extension.

Remarks

Use this property to specify a default filename extension, such as .txt or .doc.

When a file with no extension is saved, the extension specified by this property is automatically appended to the filename.

Data Type

String

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

DefaultPercentBasis Property

See Also Example Applies To

Returns the default axis percentage basis for the chart.

Syntax

object.**DefaultPercentBasis**

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

Return Value

An integer that specifies the default axis percentage basis. Possible return values are VtChPercentAxisBasis constants.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DefaultType Property (Data Control)

See Also Example Applies To

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. Also, if you select "1 - UseODBC" in the **DefaultType** property in the property sheet, you must also specify an ODBC connect string in the **Connect** property box. To do this, select Text, then enter the ODBC connect string. For more information on ODBC connect strings, see the Visual Basic *Data Access Guide*.

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataGrid Control

Visual Studio 6.0

DefaultValue Property (DataGrid Control)

See Also Example Applies To

Sets the default value for new column data.

Syntax

object.**DefaultValue** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Variant expression containing the default value for the specified column.

Remarks

The **DataGrid** control does not use this property itself, but provides it as a placeholder for you to associate default values with columns in a **DataGrid** control. This property can be used as a tag for a column. Arbitrary values can be stored and retrieved later.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataGrid Control

Visual Studio 6.0

DefColWidth Property

See Also Example Applies To

Returns or sets a value indicating the default column width for all columns in the **DataGrid** control.

Syntax

object.**DefColWidth** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer based on the scale mode of the control.

Remarks

If you set the **DefColWidth** property to 0, the control automatically sizes all columns based on either the width of the column heading or the **Size** property setting of the underlying field, whichever is larger. For example, to set the default column width of all columns to the width of the first column:

```
DataGrid1.DefColWidth = DataGrid1.Columns(0).Width
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DEFields Property

See Also Example Applies To

Returns a reference to the DEFields collection. Read-only.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DefinedSize Property

See Also Example Applies To

Returns the size of the DEField object value, as defined by the structure of the database. This property is read-only.

Note This property is different from the ADO ActualSize property, which returns the actual size of a given field within a given row.

Syntax

object.**DefinedSize** [=*value*]

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

Part	Description
object	An object expression that evaluates to an item in the Applies To list.
value	A Long value that specifies the defined size of the DEField object.

Remarks

This property corresponds to the ADO Field DefinedSize property.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DEGroupingFields Property

See Also Example Applies To

Returns a reference to the DEGroupingFields collection. Read-only.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DEParameters Property

See Also Example Applies To

Returns a reference to the DEParameters collection. Read-only.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

DepthToHeightRatio Property

See Also Example Applies To

Returns or sets the percentage of the chart height to be used as the chart depth.

Syntax

object.**DepthToHeightRatio** [= *pctg*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
pctg	Single. The chart height percentage.

Remarks

This property has an effect only on 3 dimensional chart types.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DERelationConditions Property

See Also Example Applies To

Returns a reference to the DERelationConditions collection. Read-only.

© 2017 Microsoft

Visual Basic Extensibility Reference

Description Property Example

The first of the following examples uses the **Description** property to assign a description to a particular project; the example also prints the description to verify that the assignment was successful.

The second example uses the **Description** property to return the descriptive names of the specified **Reference** objects in a particular project.

```
Application.VBE.VBProjects(1).Description = "Hot Sauce"
Debug.Print Application.VBE.VBProjects(1).Description

Debug.Print Application.VBE.VBProjects(1).References(1).Description

Debug.Print Application.VBE.VBProjects(1).References(2).Description
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic for Applications Reference

Visual Studio 6.0

Description Property

See Also Example Applies To Specifics

Returns or sets a string expression containing a descriptive string associated with an object. Read/write.

For the **Err** object, returns or sets a descriptive string associated with an error.

Remarks

The **Description** property setting consists of a short description of the error. Use this property to alert the user to an error that you either can't or don't want to handle. When generating a user-defined error, assign a short description of your error to the **Description** property. If **Description** isnt filled in, and the value of **Number** corresponds to a Visual Basic run-time error, the string returned by the **Error** function is placed in **Description** when the error is generated.

© 2017 Microsoft

Visual Basic for Applications Reference

Description Property Example

This example assigns a user-defined message to the **Description** property of the **Err** object.

```
Err.Description = "It was not possible to access an object necessary " _
& "for this operation."
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

Description Property (Button Object)

See Also Example Applies To

Returns or sets the text for a **Button** object's description, which is displayed in the Customize Toolbar dialog box.

Syntax

object.**Description** [= *string*]

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

Part	Description
object	An object expression that evaluates to a **Button** object.
string	The string displayed in the Customize Toolbar dialog box when the button is selected.

Remarks

At run time, the Customize Toolbar dialog box can be invoked either by a user double-clicking the **Toolbar** control or programmatically using the **Customize** method. In either case, when the user selects a button in the dialog box, a description of the button is displayed in the lower-left corner of the dialog box. The text for that description is set with the **Description** property.

You can set the **Description** text when you add a **Button** object, as follows:

```
Dim btnX As Button
' Add a button with the Key "save."
Set btnX = Toolbar1.Buttons.Add(,"save")
btnX.Description = "Save a file."
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: RDO Data Control

Visual Studio 6.0

Description Property (Remote Data)

See Also Example Applies To

Returns a descriptive string associated with an error.

Syntax

object.**Description**

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

Return Values

The **Description** property return value is a string expression containing a description of the error.

Remarks

When an error occurs either on the remote server, or in the ODBC interface while processing your query, an **rdoError** object is created and appended to the **rdoErrors** collection. The **rdoError** object's **Description** property returns a short description and context information about where the error occurred. This can be used to alert the user to an error that you cannot, or do not want to handle. The SQLState code is appended to the front of message, followed by a colon and a space. For example "S0021: Cannot find XXX".

© 2017 Microsoft

Visual Basic: RDO Data Control

Error Description and Number Properties Example

The following code opens a read-only ODBC cursor connection against the SQL Server "SEQUEL" and includes a simple error handler that displays the error description and number.

```
Sub MakeConnection()
Dim rdoCn As New rdoConnection
On Error GoTo CnEh
With rdoCn
    .Connect = "UID=;PWD=;Database=WorkDB;" _
        & "Server=SEQUEL;Driver={SQL Server}" _
        & "DSN='';"
    .LoginTimeout = 5
    .CursorDriver = rdUseODBC
    .EstablishConnection rdDriverNoPrompt, True
End With
AbandonCn:
Exit Sub

CnEh:
Dim er As rdoError
Dim msg as string
    Msg = "An error occured " _
    & "while opening the connection:" _
    & Err & " - " & Error & VbCr
    For Each er In rdoErrors
        Msg = Msg & er.Description  _
         & ":" & er.Number & VbCr
    Next er
    Resume AbandonCn
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

Description Property

See Also Example Applies To Specifics

Returns or sets a string expression containing a descriptive string associated with an object. For the **VBProject** object, read/write; for the **Reference** object, read-only.

Remarks

For the **VBProject** object, the **Description** property returns or sets a descriptive string associated with the active project.

For the **Reference** object, the **Description** property returns the descriptive name of the reference.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

Designer Property

See Also Example Applies To Specifics

Returns the object that enables you to access the design characteristics of a component.

Remarks

If the object has an open designer, the **Designer** property returns the open designer; otherwise a new designer is created. The designer is a characteristic of certain **VBComponent** objects. For example, when you create certain types of **VBComponent** object, a designer is created along with the object. A component can have only one designer, and it's always the same designer. The **Designer** property enables you to access a component-specific object. In some cases, such as in standard modules and class modules, a designer isn't created because that type of **VBComponent** object doesn't support a designer.

The **Designer** property returns **Nothing** if the **VBComponent** object doesn't have a designer.

© 2017 Microsoft

Visual Basic Extensibility Reference

Designer Property Example

The following example uses the **Designer** and **Count** properties to return the number of controls on a form. Note that the window containing the form must be selected. The **Designer** object is the form itself.

```
Debug.Print Application.VBE.SelectVBComponent.Designer.Controls.Count
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

DesignerID Property

See Also Example Applies To Specifics

Read-only property that returns the ProgID of an ActiveX designer.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DesignPassword Property

See Also Example Applies To

Returns or sets the password for the user name at design time.

Note This property is only saved when the **DesignSaveAuthentication** property is **True**.

Syntax

object.**DesignPassword** [=*string*]

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

Part	**Description**
object	An object expression that evaluates to an item in the Applies To list.
string	A string expression that defines the password that is used at design time.

Remarks

This property is passed to ADO by appending "Password=" and a value onto the connection string which is passed to ADO at design time.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DesignPromptBehavior Property

See Also Example Applies To

Specifies the prompting behavior by the **DEConnection** object, such as when to request logon information, at design time.

Syntax

object.**DesignPromptBehavior** [=*value*]

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

Part	**Description**
object	An object expression that evaluates to an item in the Applies To list.
value	A constant or value that specifies the prompt behavior of the DEConnection object.

Remarks

This property corresponds to the ADO Connection Prompt property.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DesignSaveAuthentication Property

See Also Example Applies To

Specifies whether the authentication information is saved with the design-time class.

Syntax

object.**DesignSaveAuthentication** [=*Boolean*]

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

Part	Description
object	An object expression that evaluates to an item in the Applies To list.
Boolean	A Boolean expression that specifies whether to save the authentication information with the design-time class. The default is **False**.

Remarks

When set to **True**, the user identification and password are saved with the project.

Note Since the password is not encrypted, for maximum security you should not specify your password to persist for design time.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DesignUserName Property

See Also Example Applies To

Returns or sets the user name that is used when the connecting to the data source at design time.

Note This property is only saved if the **DesignSaveAuthentication** property is **True**.

Syntax

object.**DesignUserName** [=*string*]

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

Part	**Description**
object	An object expression that evaluates to an item in the Applies To list.
string	A string expression that specifies the user name to use when connecting to the data source at design time.

Remarks

This property is passed to ADO by appending "User ID=" and a value onto the connection string that is passed to ADO at design time.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Multimedia MCI Control

Visual Studio 6.0

DeviceID Property (Multimedia MCI Control)

See Also Example Applies To

Specifies the device ID for the currently open MCI device. This property is not available at design time and is read-only at run time.

Syntax

[*form*.]*MMControl*.**DeviceID**[= *id%*]

Remarks

The argument *id%* is the device ID of the currently open MCI device. This ID is obtained from MCI_OPEN as a result of an Open command. If no device is open, this argument is 0.

Data Type

Integer

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DeviceName Property

See Also Example Applies To

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.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Multimedia MCI Control

Visual Studio 6.0

DeviceType Property (Multimedia MCI Control)

See Also Example Applies To

Specifies the type of MCI device to open.

Syntax

[*form*.]*MMControl*.**DeviceType**[= *device$*]

Remarks

The argument *device$* is the type of MCI device to open: AVIVideo, CDAudio, DAT, DigitalVideo, MMMovie, Other, Overlay, Scanner, Sequencer, VCR, Videodisc, or WaveAudio.

The value of this property must be set when opening simple devices (such as an audio CD that does not use files). It must also be set when opening compound MCI devices when the file-name extension does not specify the device to use.

Data Type

String

© 2017 Microsoft

Visual Basic: Multimedia MCI Control

Examples (Multimedia MCI Control)

Visual Basic Example

The following example illustrates the procedure used to open an MCI device with a compatible data file. By placing this code in the Form_Load procedure, your application can use the **Multimedia MCI** control "as is" to play, record, and rewind the file Gong.wav. To try this example, first create a form with a **Multimedia MCI** control.

```
Private Sub Form_Load ()
   ' Set properties needed by MCI to open.
   MMControl1.Notify = FALSE
   MMControl1.Wait = TRUE
   MMControl1.Shareable = FALSE
   MMControl1.DeviceType = "WaveAudio"
   MMControl1.FileName = "C:\WINDOWS\MMDATA\GONG.WAV"

   ' Open the MCI WaveAudio device.
   MMControl1.Command = "Open"
End Sub
```

To properly manage multimedia resources, you should close those MCI devices that are open before exiting your application. You can place the following statement in the Form_Unload procedure to close an open MCI device before exiting from the form containing the **Multimedia MCI** control.

```
Private Sub Form_Unload (Cancel As Integer)
   MMControl1.Command = "Close"
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Page Designer

Visual Studio 6.0

DHTMLEvent Property

See Also Example Applies To

Returns the **event** object in the Dynamic HTML object model.

Syntax

object.**DHTMLEvent**

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

Remarks

To learn the X coordinate of a mouse click, a user programming with the Dynamic HTML object model would normally have to type:

```
Document.parentWindow.event.x
```

But by using this Visual Basic shortcut, the user need only type:

```
DHTMLEvent.x
```

To cancel the bubbling of an event in Dynamic HTML, one might write:

```
Window.event.cancelBubble=1
```

-or-

```
Document.parentWindow.event.cancelBubble=1
```

However, with the **DHTMLEvent** property, a Visual Basic user need only type:

```
DHTMLEvent.cancelBubble=1
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: CommonDialog Control

Visual Studio 6.0

DialogTitle Property

See Also Example Applies To

Returns or sets the string displayed in the title bar of the dialog box.

Syntax

object.**DialogTitle** [= *title*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
title	A string expression specifying the name of the dialog box.

Remarks

Use this property to display the name of the dialog box in the title bar.

Note The **CommonDialog** control ignores the setting of the **DialogTitle** property when displaying the Color, Font, or Print dialog boxes.

The default title for an Open dialog box is Open; the default title for a Save As dialog box is Save As.

Data Type

String

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Direction Property (DEDesigner Extensibility)

See Also Example Applies To

Returns or sets the direction in which a **DEParameter** object is passed to or from a procedure.

Syntax

object.**Direction** [=*value*]

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

Part	Description
object	An object expression that evaluates to an item in the Applies To list.
value	A constant or value that specifies the direction.

Remarks

This property corresponds to the ADO Parameter Direction property.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: RDO Data Control

Visual Studio 6.0

Direction Property (Remote Data)

See Also Example Applies To

Returns or sets a value indicating how a parameter is passed to or from a procedure.

Syntax

object.**Direction** [= *value*]

The **Direction** 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** as described in Settings.

Settings

The settings for *value* are one of the following values:

Constant	Value	Description
rdParamInput	0	(Default) The parameter is used to pass information to the procedure.
rdParamInputOutput	1	The parameter is used to pass information both to and from the procedure.
rdParamOutput	2	The parameter is used to return information from the procedure as in an output parameter in SQL.
rdParamReturnValue	3	The parameter is used to return the return status value from a procedure.

Remarks

When working with stored procedures, and parameter queries you should identify those parameters that are to be managed by RDO on your behalf but only when using drivers that do not automatically detect parameter direction. A parameterized query can take virtually any number of input arguments each of these need to be marked when you create your query.

Generally, your query returns a set of rows that meet the requirements established in the query based on the parameters you provide at runtime. However, when working with stored procedures, another aspect is exposed. Stored procedures return information using row sets, return status, and output parameters. Because of this, each parameter returned by your stored procedure must be marked when creating your query.

The **Direction** property determines whether the parameter is an input parameter, output parameter, or both or if the parameter is the return value from the procedure.

Note When first addressing the rdoParameter object to set the Direction property you might trip a trappable error if the rdoParameters collection could not be created. Generally this is due to syntax errors in the query or other problems that prevented RDO from creating the collection.

Some ODBC drivers do not provide information on the direction of parameters to a SELECT statement or procedure call so all parameter directions default to **rdParamInput.** In these cases, it is necessary to set the direction in code prior to executing the query.

Note The Microsoft SQL Server 6.x driver automatically sets the **Direction** property for all procedure parameters so you should not have to set the **Direction** property for any of your queries' parameters.

The **Direction** property is associated with the **rdoParameter** object but it is generally unnecessary to address the **rdoParameter** object itself as it is the default collection of the **rdoQuery** object as shown in the examples below.

For example, the following procedure returns a value from a stored procedure:

```
{? = call sp_test}
```

This call produces one parameter the return value. It is necessary to set the direction of this parameter to **rdParamOutput** or **rdParamReturnValue** before executing the prepared statement. For example:

```
Dim my_statement As rdoQuery
Set my_statement = someRdoConnection.CreateQuery _
   ("MyPs", "{? = call sp_testprocedure }", ...)
my_statement.rdoParameters(0).Direction =  _
   rdParamReturnValue
my_statement.Execute
Print my_statement.rdoParameters(0)
```

You need to set all parameter directions except **rdParamInput** before accessing or setting the values of the parameters and before executing the **rdoQuery**.

You should use **rdParamReturnValue** for return values, but you can use **rdParamOutput** where **rdParamReturnValue** is not supported.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

DisabledImageList Property

See Also Example Applies To

Returns or sets the **ImageList** control to be used for disabled images.

Syntax

object.**DisabledImageList** [= *imageList*]

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 for disabled images.

Remarks

The **Button** object can display only one image for each button. At run time, it first determines how the button should be drawn (i.e., normal, 'hot' or disabled) and then uses the image from the appropriate image list (**ImageList**, **DisabledImageList** or **HotImageList**) using the sole **Image** property as the key. It is important to understand that related images in each of the three image lists must be consistently named so that the **Toolbar** control pulls the correct ones. For example, if a particular button is making use of all three image types, then each of the three images must be defined in their respective image lists to have either the same **Index** as the other two or the same **Key**.

Setting the Toolbar control's **Enabled** property to **False** will not cause the images contained by the **DisabledImageList** to display. Instead, set the **Button** object's **Enabled** property is set to **False** to display an image from the **DisabledImageList**. To disable all buttons, use the code shown below:

```
Private Sub DisableAllButtons
   Dim b As Button
   For Each b In Toolbar1.Buttons
      b.Enabled = False
   Next
End Sub
```

Note The images in the **ImageList** control assigned to the **DisabledImageList** property must be the same size as the images in the **ImageList** control assigned to the **ImageList** property.

© 2017 Microsoft

Visual Basic: Windows Controls

DisabledImageList, HotImageList, ImageList Properties Example

The following example demonstrates the use of the **DisableImageList**, **HotImageList**, and **ImageList** properties. Notice that all three image lists use the same **Key** property. The example assumes that three **ImageList** controls named imgNormal, imgHot, and imgDisabled have been placed on a form.

```
Private Sub Form_Load()
   Dim str As String
   str = "D:\VB\Icons\Misc\" ' Change to match your graphics location.
   imgNormal.ListImages.Add Key:="face", _
   Picture:=LoadPicture(str & "Face02.ico")
   imgHot.ListImages.Add Key:="face", _
   Picture:=LoadPicture(str & "Face03.ico")
   imgDisabled.ListImages.Add Key:="face", _
   Picture:=LoadPicture(str & "Face01.ico")

   ' Set the ImageList, DisableImageList, and HotImageList properties.
   With Toolbar1
      .ImageList = imgNormal
      .DisabledImageList = imgDisabled
      .HotImageList = imgHot
      .Buttons.Add Caption:="Sad", Image:="face", Style:=tbrCheck
      .Buttons.Add Caption:="Happy", Image:="face", Style:=tbrCheck
      .Buttons.Add Caption:="Face", Image:="face", Style:=tbrCheck

      .Buttons(1).Enabled = False
      .Buttons(3).Value = tbrPressed
   End With
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DisabledPicture Property

See Also Example Applies To

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.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: RichTextBox Control

Visual Studio 6.0

DisableNoScroll Property

See Also Example Applies To

Returns or sets a value that determines whether scroll bars in the **RichTextBox** control are disabled.

Syntax

object.**DisableNoScroll** [= *boolean*]

The **DisableNoScroll** 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 scroll bars are enabled, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
False	(Default) Scroll bars appear normally when displayed.
True	Scroll bars appear dimmed when displayed.

Remarks

The **DisableNoScroll** property is ignored when the **ScrollBars** property is set to 0 (None). However, when **ScrollBars** is set to 1 (Horizontal), 2 (Vertical), or 3 (Both), individual scroll bars are disabled when there are too few lines of text to scroll vertically or too few characters of text to scroll horizontally in the **RichTextBox** control.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DisableWarnings Property

See Also Example Applies To

Returns or sets whether warning messages display when using the Data Environment designer.

Note When you change this property for one DataEnvironment object, all DataEnvironment objects on your system automatically incur the change.

Syntax

object.**DisableWarnings** [=*Boolean*]

Parameters

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

Part	Description
object	An object expression that evaluates to an item in the Applies To list.
Boolean	A Boolean expression that specifies whether warning messages are disabled or enabled.

Remarks

This property corresponds to the **Disable Warnings** check box in the **Options** dialog box and is specific to your system. When adding and removing objects using the extensibility object, you may want to programmatically set this property so no user interfaces appear.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DisplayAsDefault Property

See Also Example Applies To

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.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DisplayBind Property

See Also Example Applies To

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.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

DisplayModel Property

See Also Example Applies To

Returns or sets the display model used by the system.

Syntax

object.**DisplayModel As vbext_VBADisplayModel**

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

Settings

The settings for **vbext_VBADisplayModel** are:

Constant	Value	Description
vbext_dm_SDI	0	(Default) Display model is SDI (Single Document Interface).
vbext_dm_MDI	1	Display model is MDI (Multiple Document Interface).

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DisplayName Property

See Also Example Applies To

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.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DisplayName Property (DEDesigner Extensibility)

See Also Example Applies To

Returns a description of the connection, which includes the database and user name, that is displayed in the Data View.

Syntax

object.**DisplayName** [=*string*]

Parameters

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

Part	Description
object	An object expression that evaluates to an item in the Applies To list.
string	A string expression that specifies the database and user name that is used by Data View.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DisplayType Property

See Also Example Applies To

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	Description
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.

This documentation is archived and is not being maintained.

Visual Basic: RichTextBox Control

Visual Studio 6.0

DisplayType Property (OLEObject Object)

See Also Example Applies To

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	Description
rtfDisplayContent	0	Content. When the **RichTextBox** control contains an object, the object's data is displayed in the control.
rtfDisplayIcon	1	Icon. When the **RichTextBox** control contains an object, the object's icon is displayed in the control.

Remarks

After adding the object to the **OLEObjects** Collection, use the **DisplayType** property to change how that object is displayed.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataGrid Control

Visual Studio 6.0

DividerStyle Property

See Also Example Applies To

Returns or sets a value specifying the style of border drawn on the right edge of the specified column of a **DataGrid** control.

Syntax

object.**DividerStyle** [= *value*]

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

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

Settings

The settings for *value* are:

Constant	Value	Description
dbgNoDividers	0	No divider
dbgBlackLine	1	Black line
dbgDarkGrayLine	2	(Default) Dark gray line
dbgRaised	3	Raised
dbgInset	4	Inset
dbgUserForeColor	5	Divider is drawn using the color set by the **ForeColor** property
dbgLightGrayLine	6	Light gray line

Remarks

The **DividerStyle** property doesn't affect whether the column can be resized by dragging it or not. When the border is either raised or inset, the colors used are set by Microsoft Windows.

© 2017 Microsoft

Visual Basic: DataGrid Control

DividerStyle Property Example

This example changes the column divider style when the user clicks a heading.

```
Private Sub DataGrid1_HeadClick (ColIndex As Long)
   If DataGrid1.Columns(ColIndex).DividerStyle <> 5 Then
      DataGrid1.Columns(ColIndex).DividerStyle = _
       DataGrid1.Columns(ColIndex).DividerStyle + 1
   Else
      DataGrid1.Columns(ColIndex).DividerStyle = 0
   End If
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

DivisionsPerLabel Property

See Also Example Applies To

Returns or sets the number of divisions to skip between labels.

Syntax

object.**DivisionsPerLabel** [= *num*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
num	Integer. An integer representing the number of divisions.

Remarks

If this property is set, the object's **Auto** property is automatically set to **False**.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

DivisionsPerTick Property

See Also Example Applies To

Returns or sets the number of divisions to skip between major tick marks.

Syntax

object.**DivisionsPerTick** [= *num*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
num	Integer. An integer representing the number of divisions.

Remarks

If this property is set, the object's **Auto** property is automatically set to **False**.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Internet Control

Visual Studio 6.0

Document Property

See Also Example Applies To

Returns or sets the file or document that will be used with the **Execute** method. If this property is not specified, the default document from the server will be returned; for write operations, an error occurs if no document is specified.

Syntax

object.**Document** = *string*

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	The name of the file or document to be used with the **Execute** method.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Page Designer

Visual Studio 6.0

Document Property (DHTMLPageDesigner)

See Also Example Applies To

Returns the **Document** object from the Dynamic HTML object model, which represents the HTML page you view in the browser.

Syntax

object.**Document**

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

Remarks

You use this object to respond to events that occur on the page, and to set the pages properties.

Any of the collections available from the Dynamic HTML **Document** object are available.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

DoSetCursor Property

See Also Example Applies To

Returns or sets a value that indicates whether or not the cursor can be set by a chart. The **DoSetCursor** property determines whether or not the application can control what the mouse pointer looks like.

Syntax

object.**DoSetCursor** [= *boolean*]

The **DoSetCursor** 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 custom text is used, as described in Settings.

Settings

The settings for *boolean* are:

Setting	**Description**
True	(Default) The application can control the mouse pointer appearance.
False	The application cannot control the mouse pointer appearance.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MAPI Controls

Visual Studio 6.0

DownloadMail Property

See Also Example Applies To

Specifies when new messages are downloaded from the mail server for the designated user.

Syntax

object.**DownloadMail** [= *value*]

The **DownloadMail** 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 when new mail is downloaded, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	(Default) All new messages from the mail server are forced to the user's Inbox during the sign-on process.
False	New messages on the server are *not* forced to the user's Inbox immediately, but are downloaded at the time interval set by the user.

Remarks

This property can be set to **True** when you want to access the user's complete set of messages when signing on. However, processing time may increase as a result.

Data Type

Boolean

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DownPicture Property

See Also Example Applies To

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** controls **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.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DragIcon Property

See Also Example Applies To

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.

© 2017 Microsoft

Visual Basic Reference

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
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DragMode Property

See Also Example Applies To

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.

© 2017 Microsoft

Visual Basic Reference

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
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DrawMode Property

See Also Example Applies To

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.

© 2017 Microsoft

Visual Basic Reference

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
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

DrawMode Property (MSChart)

See Also Example Applies To

Returns or sets a value that determines when and how a chart is repainted.

Syntax

object.**DrawMode** [= *mode*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
mode	Integer. A value that determines how and when the chart will be redrawn, as shown in Settings.

Settings

The settings for *mode* are:

Constant	Value	Description
VtChDrawModeDraw	0	Draws directly to the display device.
VtChDrawModeBlit	1	Blits an offscreen drawing to the display device.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DrawStyle Property

See Also Example Applies To

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.

© 2017 Microsoft

Visual Basic Reference

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
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DrawWidth Property

See Also Example Applies To

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.

© 2017 Microsoft

Visual Basic Reference

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
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Drive Property

See Also Example Applies To

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.

© 2017 Microsoft

Visual Basic Reference

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
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic for Applications Reference

Visual Studio 6.0

DriveLetter Property

See Also Example Applies To Specifics

Description

Returns the drive letter of a physical local drive or a network share. Read-only.

Syntax

object.**DriveLetter**

The *object* is always a **Drive** object.

Remarks

The **DriveLetter** property returns a zero-length string ("") if the specified drive is not associated with a drive letter, for example, a network share that has not been mapped to a drive letter.

The following code illustrates the use of the **DriveLetter** property:

```
Sub ShowDriveLetter(drvPath)
    Dim fs, d, s
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set d = fs.GetDrive(fs.GetDriveName(drvPath))
    s = "Drive " & d.DriveLetter & ": - "
    s = s & d.VolumeName  & vbCrLf
    s = s & "Free Space: " & FormatNumber(d.FreeSpace/1024, 0)
    s = s & " Kbytes"
    MsgBox s
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

DriverName Property

See Also Example Applies To

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.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic for Applications Reference

Visual Studio 6.0

Drives Property

See Also Example Applies To Specifics

Description

Returns a **Drives** collection consisting of all **Drive** objects available on the local machine.

Syntax

object.**Drives**

The *object* is always a **FileSystemObject**.

Remarks

Removable-media drives need not have media inserted for them to appear in the **Drives** collection.

You can iterate the members of the **Drives** collection using a **For Each...Next** construct as illustrated in the following code:

```
Sub ShowDriveList
    Dim fs, d, dc, s, n
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set dc = fs.Drives
    For Each d in dc
        s = s & d.DriveLetter & " - "
        If d.DriveType = 3 Then
            n = d.ShareName
        Else
            n = d.VolumeName
        End If
        s = s & n & vbCrLf
    Next
    MsgBox s
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic for Applications Reference

Visual Studio 6.0

DriveType Property

See Also Example Applies To Specifics

Description

Returns a value indicating the type of a specified drive.

Syntax

object.**DriveType**

The *object* is always a **Drive** object.

Remarks

The following code illustrates the use of the **DriveType** property:

```
Sub ShowDriveType(drvpath)
    Dim fs, d, s, t
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set d = fs.GetDrive(drvpath)
    Select Case d.DriveType
        Case 0: t = "Unknown"
        Case 1: t = "Removable"
        Case 2: t = "Fixed"
        Case 3: t = "Network"
        Case 4: t = "CD-ROM"
        Case 5: t = "RAM Disk"
    End Select
    s = "Drive " & d.DriveLetter & ": - " & t
    MsgBox s
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

DropHighlight Property (ListView, TreeView Controls)

See Also Example Applies To

Returns or sets a reference to a **Node** or **ListItem** object that is highlighted with the system highlight color when the cursor moves over it.

Syntax

```
object.DropHighlight [ = value]
```

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A **Node** or **ListItem** object.

Remarks

The **DropHighlight** property is typically used in combination with the **HitTest** method in drag-and-drop operations. As the cursor is dragged over a **ListItem** or **Node** object, the **HitTest** method returns a reference to any object it is dragged over. In turn, the **DropHighlight** property is set to the hit object, and simultaneously returns a reference to that object. The **DropHighlight** property then highlights the hit object with the system highlight color. The following code sets the **DropHighlight** property to the object hit with the **HitTest** method.

```
Private Sub TreeView1_DragOver _
(Source As Control, X As Single, Y As Single, State As Integer)
   Set TreeView1.DropHighlight = TreeView1.HitTest(X,Y)
End Sub
```

Subsequently, you can use the **DropHighlight** property in the DragDrop event to return a reference to the last object the source control was dropped over, as shown in the following code:

```
Private Sub TreeView1_DragDrop _
(Source As Control, x As Single, y As Single)
   ' DropHighlight returns a reference to object drop occurred over.
   Me.Caption = TreeView1.DropHighlight.Text
   ' To release the DropHighlight reference, set it to Nothing.
```

```
    Set TreeView1.DropHighlight = Nothing
End Sub
```

Note that in the preceding example, the **DropHighlight** property is set to Nothing after the procedure is completed. This must be done to release the highlight effect.

© 2017 Microsoft

Visual Basic: Windows Controls

DropHighlight Property Example

This example adds several **Node** objects to a **TreeView** control. After you select a **Node** object, you can drag it to any other **Node**. To try the example, place **TreeView** and **ImageList** controls on a form and paste the code into the form's Declaration section. Run the example and drag **Node** objects around to see the result.

Note The graphics files in the code below can be found on Disk 1 of the Visual Basic or Visual Studio CDs, in the Common\Graphics directory. Change the path in the code, or copy the graphics files to your hard disk before running the code.

```
' Declare global variables.
Dim indrag As Boolean ' Flag that signals a Drag Drop operation.
Dim nodX As Object ' Item that is being dragged.

Private Sub Form_Load()
   ' Load a bitmap into an Imagelist control.
   Dim imgX As ListImage
   Dim BitmapPath As String
   BitmapPath = "icons\mail\mail01a.ico" ' Change to a valid path.
   Set imgX = imagelist1.ListImages.Add(, , LoadPicture(BitmapPath))

   ' Initialize TreeView control and create several nodes.
   TreeView1.ImageList = imagelist1
   Dim nodX As Node   ' Create a tree.
   Set nodX = TreeView1.Nodes.Add(, , , "Parent1", 1)
   Set nodX = TreeView1.Nodes.Add(, , , "Parent2", 1)
   Set nodX = TreeView1.Nodes.Add(1, tvwChild, , "Child 1", 1)
   Set nodX = TreeView1.Nodes.Add(1, tvwChild, , "Child 2", 1)
   Set nodX = TreeView1.Nodes.Add(2, tvwChild, , "Child 3", 1)
   Set nodX = TreeView1.Nodes.Add(2, tvwChild, , "Child 4", 1)
   Set nodX = TreeView1.Nodes.Add(3, tvwChild, , "Child 5", 1)
   nodX.EnsureVisible ' Expand tree to show all nodes.
End Sub

Private Sub TreeView1_MouseDown_
(Button As Integer, Shift As Integer, x As Single, y As Single)
   Set nodX = TreeView1.SelectedItem ' Set the item being dragged.
End Sub

Private Sub TreeView1_MouseMove _
(Button As Integer, Shift As Integer, x As Single, y As Single)
   If Button = vbLeftButton Then ' Signal a Drag operation.
      indrag = True ' Set the flag to true.
      ' Set the drag icon with the CreateDragImage method.
      TreeView1.DragIcon = TreeView1.SelectedItem.CreateDragImage
      TreeView1.Drag vbBeginDrag ' Drag operation.
   End If
End Sub

Private Sub TreeView1_DragDrop_
(Source As Control, x As Single, y As Single)
   If TreeView1.DropHighlight Is Nothing Then
      Set TreeView1.DropHighlight = Nothing
      indrag = False
      Exit Sub
   Else
      If nodX = TreeView1.DropHighlight Then Exit Sub
      Cls
      Print nodX.Text & " dropped on " & TreeView1.DropHighlight.Text
      Set TreeView1.DropHighlight = Nothing
      indrag = False
   End If
End Sub

Private Sub TreeView1_DragOver(Source As Control, x As Single, y As Single, State As Integer)
   If indrag = True Then
      ' Set DropHighlight to the mouse's coordinates.
      Set TreeView1.DropHighlight = TreeView1.HitTest(x, y)
   End If
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MSComm Control

Visual Studio 6.0

DSRHolding Property

See Also Example Applies To

Determines the state of the Data Set Ready (DSR) line. Typically, the Data Set Ready signal is sent by a modem to its attached computer to indicate that it is ready to operate. This property is not available at design time and is read-only at run time.

Syntax

object.**DSRHolding**

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

The **DSRHolding** property returns the following values:

Value	Description
True	Data Set Ready line high
False	Data Set Ready line low

Remarks

This property is useful when writing a Data Set Ready/Data Terminal Ready handshaking routine for a Data Terminal Equipment (DTE) machine.

Data Type

Boolean

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MSComm Control

Visual Studio 6.0

DTREnable Property

See Also Example Applies To

Determines whether to enable the Data Terminal Ready (DTR) line during communications. Typically, the Data Terminal Ready signal is sent by a computer to its modem to indicate that the computer is ready to accept incoming transmission.

Syntax

object.**DTREnable**[= *value*]

The **DTREnable** 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 to enable the Data Terminal Ready (DTR) line, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	Enable the Data Terminal Ready line.
False	(Default) Disable the Data Terminal Ready line.

Remarks

When **DTREnable** is set to **True**, the Data Terminal Ready line is set to high (on) when the port is opened, and low (off) when the port is closed. When **DTREnable** is set to **False**, the Data Terminal Ready always remains low.

Note In most cases, setting the Data Terminal Ready line to low hangs up the telephone.

Data Type

Boolean

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Duplex Property

See Also Example Applies To

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.

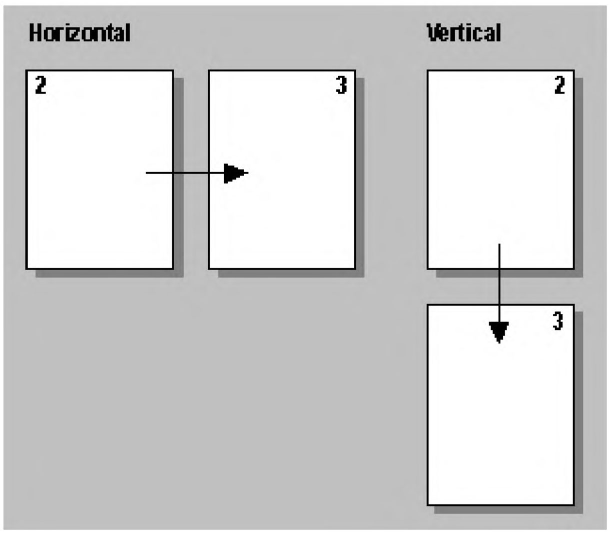
Settings

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.

© 2017 Microsoft