Using DocInput and DocOutput Objects

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmscDocInputOverviewsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmscDocInputOverviewsX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmscDocInputOverviewsS"}

The ActiveX™ Controls include several controls which allow you to connect to external servers (the Internet) and download files. To facilitate the transfer of data, the **DocInput** and the **DocOutput** objects have been created. These objects allow you to stream data into or out of controls through the DocInput and DocOutput events. Using the various properties and methods of these objects, you can determine the state of the data stream and direct it accordingly.

Possible Uses

The **DocInput** and **DocOutput** objects can be used in the following scenarios:

- To stream a message from a news server into a container.
- To stream a file retrieved from an HTTP site directly to a remote FTP site.
- To post a message to a news server.

Scenario: Use the DocOutput Object to Retrieve an NNTP Document

A common task is to download messages from a remote news server. The steps to downloading a message using the **DocOutput** object are as follows:

1. Invoke the **GetArticleByArticleNumber** method.
2. The DocOutput event occurs with the **DocOutput** object as an argument
3. The **DocOutput** passes a reference to a **DocOutput** object.
4. Determine the state of the data stream with the **DocOutput** object's **State** property.

Setup

The following code assumes that you have successfully logged on to a news server, entered a news group, and retrieved a list of all messages and their numbers. For more information on using the **NNTP** control, see "Using the **NNTP** Client control." Additionally the following controls are used:

1. **NNTP** control named "NNTP1."
2. **TextBox** control named "txtArticleNumber."

Invoke the GetArticleByArticleNumber Method

The code below invokes the **GetArticleByNumber** method to begin retrieving a message.

```
Sub GetArticleByNumber_Click()
   NNTP1.GetArticleByArticleNumber txtArticleNumber
End Sub
```

The DocOutput Event Occurs with the DocOutput Object as an Argument

If the command is successful, the server responds by sending back the article message. In turn, the **NNTP** control's DocOutput event occurs. The DocOutput event passes a reference to the **DocOutput** object.

Route the Data Stream Using the DocOutput Object's State Property

The **DocOutput** object has a **State** property which indicates the state of the data stream. The code below uses the **Select** statement with **State** property constants to determine how to route the data stream. When the transfer begins (**icDocBegin**), a file is opened for input. As data arrives (**icDocData**), it is routed into the file. When the transfer ends (**icDoc**, the file is closed.

```
Private NNTP1_DocInput(DocInput As DocInput)
```

```
Dim vtData As Variant' Variable for data.

Select Case DocInput.State
   Case icDocNone ' No transfer in progress.
      Exit Sub
   Case icDocBegin
      ' Open a file for input.
      Open "messages.txt" For Input As #1
   Case icDocHeaders ' Document headers are being
                     ' transferred.
      ' Add the Header to the DocHeaders collection.
   Case icDocData ' One block of data is
                  ' transferred.
      ' Use the Getdata Method to retrieve the data
      DocInput.GetData vtData, vbByte + vbArray
   Case icError
      ' If it's an error, use the icErrors collection.
      MsgBox icErrors.Description
   Case icDocEnd

End Case

End Sub
```

The code above demonstrates the basics of getting data from a control that uses the **DocOutput** object.

Scenario 2: Route Data to a File Using the DocLink Property

If the **DocOutput.DocLink** property is assigned to a **DocInput.DocLink** property, data is transferred between the objects. If the **DocLink** property is not assigned, no data linking occurs, but data is available via an output file and/or data streaming.

All three forms of output can be used in any combination: an output file (**FileName** property), data linking (**DocLink** property), and data streaming (DocOutput event).

To stream data directly from one control to another, you can bypass the **DocOutput** object entirely by using the **DocLink** property. The steps required to do this are:

1. Assign the **DocLink** property before invoking the **GetDoc** method.
2. Invoke the **GetDoc** method.

Assign the DocLink Property Before Invoking the GetDoc Method

The **DocInput** and the **DocOutput** objects both have a **DocLink** property. You can use this property to stream data directly from one control's **DocInput** object to another control's **DocOutput** object.

To do this, assign the **DocLink** property of a **DocOutput** object to the **DocLink** property of a **DocInput** object before invoking the **GetDoc** method. This causes data output from the **DocOutput** object to be used as the data input for the **DocInput** object. In the Visual Basic code below, the **DocLink** property of an **FTP** control is set to the **DocLink** property of an **HTTP** control:

```
ftp1.DocInput.DocLink = HTTP1.DocOutput.DocLink
```

Using the icError Object and icErrors Collection

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

The **icError** Object and the **icErrors** collection are provided to store and access errors that originate from a network. Use the **Errors** property to access the **icErrors** collection.

Possible Uses

You can use the **icErrors** collection in the following scenarios:

- To retrieve all errors that occur as a document is retrieved using the **GetDoc** method.
- To retrieve all errors that occur when a message is sent using the **SendDoc** method.

Scenario: List All Errors that Occur Retrieving a Document with the FTP Control

When retrieving a document using the **FTP** control, a variety of errors can occur. To process the errors, use the **icErrors** collection. The steps to do this are as follows:

1. With the **FTP** Client control, invoke the **GetFile** method.
2. Use the **Select Case** statement in the DocOutput event to examine the **State** property of the **DocOutput** object.
3. If the **State** property is icError then iterate through the **icErrors** collection of the **FTP** control.

Setup

The scenario presumes you are using the **FTP** control and have successfully connected to a remote FTP site. For more information on doing this, see "Using the FTP Client Control." The Visual Basic code also assumes the presence of the following controls:

- **FTP** Client control named "FTP1."
- **CommandButton** control named "cmdGetFile."
- **TextBox** control named "txtRemote."
- **TextBox** control named "txtLocal."
- **TextBox** control named "txtErrors."

With the FTP Client Control, Invoke the GetFile Method

To retrieve a file from a remote server, you can use the **GetFile** method. This method requires two arguments, the name of the remote file, and the name of the file as it will appear on the local machine. The code below invokes the **GetFile** method using the contents of two **TextBox** controls to supply the argument.

```
Private Sub cmdGetFile_Click()
   FTP1.GetFile txtRemote, txtLocal
End Sub
```

Use the Select Case Statement in the DocOutput Event to Examine the State Property of the DocOutput Object

In response to the **GetFile** method, the DocOutput event occurs. Each occurrence of the DocOutput event passes a reference to the **DocOutput** object. You can examine the **State** property of the **DocOutput** object to determine what to do with the data associated with the object. For example, if the **State** property is **icError** (4), examine each **icError** object in the **icErrors** collection.

The code below examines the **State** property of the **DocOutput** object using the **Select Case** statement. If the **State** is **icError**, code to handle the error would be placed under the **Case** statement.

```
FTP1_DocOutput(ByVal DocOutput As DocOutput)

   Select Case DocOutput.State
      Case icDocNone
         ' No document, so do nothing.

      Case icDocError
         ' Handle errors here.
   End Select

End Sub
```

If the DocOutput Sate is icError then Iterate Through the icErrors Collection of the FTP Control

The **icErrors** collection can be referenced through the **Errors** property of the **FTP** control. To handle the error, the code below uses the **For Each** statement to iterate through each member of the **icErrors** collection.

```
FTP1_DocOutput(ByVal DocOutput As DocOutput)

   Select Case DocOutput.State
      Case icDocNone
         ' No document, so do nothing.

      Case icDocError
         Dim errCol As icErrors ' icErrors variable.
         Set errCol = FTP1.Errors ' Set the variable.

         ' Iterate through the collection and place
         ' results in a textbox control.
         For Each icError In errCol
            txtErrors = txtErrors & icError.Code & _
            ":" & icError.Description & vbCrLf
         Next
      End Select
End Sub
```

Source Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproSourcePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSourcePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSourcePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSourcePropertyS"}

The **vbObject** that the most recent error applies to, or **vbEmpty**. Implementation is OCX-dependent. Unless specified by the OCX Control documentation, the value for **Source** is **vbEmpty**. Read-only and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**Source**
Visual FoxPro	*Object*.Source

Return Values

Development Tool	Default Value	Data Type
Microsoft Access and Visual Basic	None	Variant
Visual FoxPro		

Suspended Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproSuspendedPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSuspendedPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSuspendedPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSuspendedPropertyS"}

Returns a value that indicates whether a document transfer is currently suspended or not.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**Suspended**
Visual FoxPro	*Object*.Suspended

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

The **Suspended** property settings are:

Argument	Description
True	The transfer is suspended.
False	The transer is not suspended.

Data Type

Boolean.

Remarks

The transfer is suspended if the **Suspend** method of the **DocInput** object is called.

icError Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjErrorObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjErrorObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjErrorObjectP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjErrorObjectM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjErrorObjectE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjErrorObjectS"}

An **icError** object contains error messages.

icErrors Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcolErrorsCollectionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbcolErrorsCollectionX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbcolErrorsCollectionP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbcolErrorsCollectionM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbcolErrorsCollectionE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcolErrorsCollectionS"}

The **icErrors** collection is used to access errors generated by the last error condition. Some common items in the collection are Protocol error and Transport error.

A protocol error provides general error information at a protocol level. A transport error gives specific detail (where applicable) of the last error that occurred in the transport layer. Protocol and transport don't contain any data if there is no error of that type. Once an error is processed, the collection can be cleared by the **Clear** method, which also resets the **Source** property.

Code Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproCodePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCodePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCodePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCodePropertyS"}

Returns the Integer error code for the given error type. Read-only and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**Code**
Visual FoxPro	*Object*.Code

Return Values

Development Tool	Default Value	Data Type
Microsoft Access and Visual Basic	0	Long
Visual FoxPro	0	Numeric

Description Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDescriptionPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDescriptionPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDescriptionPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDescriptionPropertyS"}

Returns the text description of the error. Read-only and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**Description**
Visual FoxPro	*Object*.Description

Return Values

Development Tool	Default Value	Data Type
Microsoft Access and Visual Basic	Empty string	String
Visual FoxPro	Empty string	Character

Type Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproTypePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTypePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTypePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTypePropertyS"}

Returns a string label for the type of error, with standard (predefined) labels such as "protocol" and "transport". Individual controls can have additional Type labels. Read-only and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object.***Type**
Visual FoxPro	*Object*.Type

Return Values

Development Tool	Default Value	Data Type
Microsoft Access and Visual Basic	Empty string	String
Visual FoxPro	Empty string	Character

DocHeaders Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcolDocHeadersCollectionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbcolDocHeadersCollectionX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbcolDocHeadersCollectionP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbcolDocHeadersCollectionM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbcolDocHeadersCollectionE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcolDocHeadersCollectionS"}

A **DocHeaders** collection contains a collection of **DocHeader** objects.

Remarks

You commonly add several **DocHeader** objects to a collection before invoking the **SendDoc** method. The most effecient method of doing this is to first create a Docheaders variable, instantiate the variable (using the **Set** statement), then add objects to the collection. This is demonstrated in the code below:

```
Dim docHeads As DocHeaders ' Create variable.
Set docHeads = New DocHeaders ' Instantiate variable.
With DocHeads
   .Add "From", "jamesDean@anycom.com"
   .Add "To", "marilyn@nothercom.com"
   .Add "Subject", "Hello"
End With
```

Add Method (DocHeaders Collection)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthAddDocHeaderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthAddDocHeaderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthAddDocHeaderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthAddDocHeaderS"}

Adds a new **DocHeader** object to the **DocHeaders** collection. The Name and Value arguments are converted to type String and become the **Name** and **Value** properties of the **DocHeader** object.

Return Value

Void

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**Add** *Name, Value*
Visual FoxPro	*Object*.Add(*Name, Value)*

Arguments

Development Tool	Argument	Data Type	Description
Microsoft Access and Visual Basic	*Name*	Variant	Attribute name.
	Value	Variant	Value for the specified attribute name.
Visual FoxPro	*Name*		Attribute name.
	Value		Value for the specified attribute name.

Remarks

To add a **DocHeader** object to the **DocHeaders** collection, create an object variable of type DocHeaders, initialize the variable, then add **DocHeader** objects:

```
Dim dchX As DocHeaders
Set dchX As New DocHeaders
dchX.Add("Subject", "Development")
dchX.Add("Name", "Joe@MyCom.Com")
```

Item Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthICPItemC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthICPItemX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthICPItemA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthICPItemS"}

Returns a specific member of a collection. The **Item** method is the default method for a collection.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**Item(***index***)**
Visual FoxPro	*Object*.Item(*index*)

Arguments

Development Tool	Argument	Data Type	Description
Microsoft Access and Visual Basic	*Index*	Variant	Identifies the item in the collection.
Visual FoxPro			

Clear Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthICPClearC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthICPClearX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthICPClearA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthICPClearS"}

Clears all objects in a collection.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**Clear**
Visual FoxPro	*Object*.Clear( )

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

Remarks

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

Remove Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthICPRemoveC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthICPRemoveX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthICPRemoveA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthICPRemoveS"}

Removes a member from a collection object.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object.***Remove** *index*
Visual FoxPro	*Object.*Remove(*nIndex*)

Arguments

Development Tool	Argument	Data Type	Description
Microsoft Access and Visual Basic	*index*	Integer	Required. An expression that specifies the position of a member of the colllection.
Visual FoxPro	*nIndex*	Numeric	Required. A numeric expression that specifies the position of a member of the colllection.

Count Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproICPCountC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproICPCountX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproICPCountA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproICPCountS"}

Returns the number of objects in a collection. Read-only and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object.***Count**
Visual FoxPro	*Object*.Count

Return Values

Development Tool	Default Value	Data Type
Microsoft Access and Visual Basic	None	Long
Visual FoxPro	None	Numeric

DocHeader Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjDocHeaderObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjDocHeaderObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjDocHeaderObjectP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjDocHeaderObjectM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjDocHeaderObjectE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjDocHeaderObjectS"}

A **DocHeader** object contains a **Name** property and a **Value** property which together form a single MIME (Multipurpose Internet Mail Extensions) header.

Remarks

The MIME protocol was established to accommodate the transfer of multi-part mail messages. For example, a document may consist of both an HTML text section and a sound file in the .WAV format. A **DocHeader** object contains the information on what kind of data is contained in a particular mail message.. The **DocHeader** object contains just two properties, the **Name** and **Value** properties. Together, they create a single header.

The **DocHeader** object supports the following properties:

Property	Description
Name	The item name or MIME header label (not including the colon character). **DocHeader** objects represent individual name and value pairs in MIME headers.
Value	The item value which in MIME headers is the text after the label, colon character, and any leading spaces.

Name Property (DocHeader Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproNamePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproNamePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproNamePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproNamePropertyS"}

Returns or sets the item name or MIME (Multipurpose Internet Mail Extensions) header label (not including the colon character). This property can be used as an identifier for items in the **DocHeaders** collection. Read/write and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**Name**
Visual FoxPro	*Object*.Name

Return Values

Development Tool	Default Value	Data Type
Microsoft Access and Visual Basic	Empty string	String
Visual FoxPro	Empty string	Character

Value Property (DocHeader Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproValueDocHeaderObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproValueDocHeaderObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproValueDocHeaderObjectA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproValueDocHeaderObjectS"}

Returns or sets the value which in MIME (Multipurpose Internet Mail Extensions) headers is the text after the label, colon character, and any leading spaces.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**Value** [= *string*]
Visual FoxPro	*Object*.Value[= *cText*]

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

Data Type

String

Text Property (DocHeaders Collection)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproTextPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTextPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTextPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTextPropertyS"}

Returns the complete text of all headers in standard MIME (Multipurpose Internet Mail Extensions) header format. Read/write and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object.***Text** [= *String*]
Visual FoxPro	*Object*.Text[= *cExpression*]

Return Values

Development Tool	Default Value	Data Type
Microsoft Access and Visual Basic	None	String
Visual FoxPro	None	Character

Remarks

The standard text format for MIME headers follows each header with a CRLF terminator, and separates the **Name** and **Value** of each header by a colon and single space character (": ").

If the **Text** property is set, all collection items will be replaced.

DocInput Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjDocInputObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjDocInputObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjDocInputObjectP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjDocInputObjectM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjDocInputObjectE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjDocInputObjectS"}

The **DocInput** object provides input information for a document being transferred. All controls with the **DocInput** property can access properties and invoke methods of the **DocInput** object. In such controls, a reference to the **DocInput** object is also passed as an argument of the DocInput event.

Remarks

The **DocInput** object provides a more powerful interface than the basic capabilities of the **RequestDoc** method for the HTML control. However, you can use the basic functions of the control without knowledge or use of the **DocInput** object.

DocOutput Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjDocOutputObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjDocOutputObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjDocOutputObjectP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjDocOutputObjectM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjDocOutputObjectE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjDocOutputObjectS"}

The **DocOutput** object provides output information for a document being transferred. All controls with the **DocInput** property can access properties and invoke methods of the **DocOutput** object. In such controls, a reference to the **DocOutput** object is also passed as an argument of the DocOutput event.

Remarks

The **DocOutput** object provides a more powerful interface than the basic capabilities of the **RequestSubmit** method of the HTML control. However, you can use the basic functions of the control without knowledge or use of the **DocInput** object.

Filename Property (DocInput, DocOutput Objects)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproFilenamePropertyDocInputObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFilenamePropertyDocInputObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFilenamePropertyDocInputObjectA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFilenamePropertyDocInputObjectS"}

Returns or sets the name of a local file containing the document to be transferred or sent.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object.***FileName** [**=** *string*]
Visual FoxPro	*Object*.FileName[**=** *cExpression*]

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

Data Type

String

Remarks

The **FileName** property can be set before calling either the **GetDoc** or **SendDoc** method in a particular control or it can be passed as an argument to the method. If it is passed as an argument, the **FileName** property is set to the argument value.

DocInput Specific Remarks

When used as a property of the **DocInput** Object, input data can only be supplied through an input file, data linking, or data streaming. Property contents determine how the data is supplied. If the **FileName** property is not empty, it is used as the input file. If the **DocLink** property is not null, data linking is used. Otherwise, data streaming via the DocInput event is used.

When the **FileName** property is set to a nonempty value, the **DocLink** property is automatically set to empty.

DocOutput Specific Remarks

When used as a property of the **DocOutput** Object, if the **FileName** property is not empty, data is appended to the file as it is transferred. If the **FileName** property is empty, no data is written to an output file. However, data is available via data linking and/or data streaming.

All three forms of output can be used in any combination: an output file (**FileName** property), data linking (**DocLink** property), and data streaming (DocOutput event).

State Property (DocInput, DocOutput Objects)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproStatePropertyDocInputDocOutputObjectsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproStatePropertyDocInputDocOutputObjectsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproStatePropertyDocInputDocOutputObjectsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproStatePropertyDocInputDocOutputObjectsS"}

Returns a value that indicates the current state of the document transfer. Read-only, and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**State**
Visual FoxPro	*Object*.State

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

Settings

The **State** property is maintained by the specific control. Each time it changes, the DocInput event is activated. The **State** property is always set to one of the values listed here.

Name	Value	Description
icDocNone	0	No transfer is in progress.
icDocBegin	1	Transfer is being initiated
icDocHeaders	2	Document headers are transferred (or requested).
icDocData	3	One block of data is transferred (or requested).
icDocError	4	An error has occurred during transfer.
icDocEnd	5	Transfer is complete (either successfully or with an error).

DocLink Property (DocOutput Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDocOutputDocLinkC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDocOutputDocLinkX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDocOutputDocLinkA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDocOutputDocLinkS"}

Returns a reference to the **DocLink** property when data linking is used. Read-only and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**DocLink**
Visual FoxPro	*Object*.DocLink

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

Remarks

The **DocInput** and the **DocOutput** objects both have a **DocLink** property. You can use this property to stream data directly from one control's **DocInput** object to another control's **DocOutput** object.

To do this, assign the **DocLink** property of a **DocOutput** object to the **DocLink** property of a **DocInput** object before invoking the **GetDoc** method. This causes data output from the **DocOutput** object to be used as the data input for the **DocInput** object. In the Visual Basic code below, the **DocLink** property of an **FTP** control is set to the **DocLink** property of an **NNTP** control:

```
Private Sub cmdFTPGetDoc_Click()
   ftp1.DocInput.DocLink = nntp1.DocOutput.DocLink
   ftp1.GetDoc
End Sub
```

If the **DocOutput.DocLink** property is assigned to a **DocInput.DocLink** property, data is transferred between objects. If the **DocLink** property is not assigned, no data linking occurs, but data is available via an output file and/or data streaming.

All three forms of output can be used in any combination: an output file (**FileName** property), data linking (**DocLink** property), and data streaming (DocOutput event).

GetData Method (DocInput, DocOutput Objects)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthDocInputGetDataC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthDocInputGetDataX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthDocInputGetDataA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthDocInputGetDataS"}

Retrieves the current block of data being transferred when the DocInput or DocOutput event occurs.

Return Value

Void

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object.***GetData** *data***,** [*type*]
Visual FoxPro	*Object.*GetData(*eData* [, *nDataType*])

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

Arguments

Development Tool	Argument	Data Type	Description
Microsoft Access and Visual Basic	*data*	Variant	Where retrieved data will be stored after the method returns successfully. If there is not enough data available for requested type, *Data* will be set to Empty.
	type	Long	Optional. Type of data to be retrieved. Set Settings below for a list of types supported.
Visual FoxPro	*eData*	Variant	Where retrieved data will be stored after the method returns successfully. If there is not enough data available for requested type, *eData* will be set to Empty.
	nDataType	Numeric	Optional. Type of data to be retrieved. Set Settings below for a list of types supported.

Settings

The settings for *type* are:

Description	Visual C++	Visual Basic Type
Byte	**VT_UI1**	**vbByte**
Integer	**VT_I2**	**vbInteger**
Long	**VT_I4**	**vbLong**
Single	**VT_R4**	**vbSingle**
Double	**VT_R8**	**vbDouble**
Currency	**VT_CY**	**vbCurrency**

Date	**VT_DATE**	**vbDate**
Boolean	**VT_BOOL**	**vbBoolean**
SCODE	**VT_ERROR**	**vbError**
String	**VT_BSTR**	**vbString**
Byte Array	**VT_ARRAY|VT_UI1**	**vbArray + vbByte**

Remarks

The **GetData** method can only be called during handling of the DocOutput event, when the **State** property is set to **icDocData** (3).

GetData Method, DocOutput Event Example

The example below uses a **Select Case** statement to determine when data is being sent through the **DocOuput** object. If the **DocOuput** object's **State** is **icDocData** (3), the **GetData** method is invoked.

```
Sub NNTP1_DocOutput(DocOutput As DocOuput)
   Dim vtData As Variant ' Data variable.

   Select Case DocOutput.State
   Case icDocBegin
      ' Open a file to write to.
      Open "messages.txt" For Output As #1
   Case icDocData
      ' Use the GetData method to retrieve data.
      DocOutput.GetData vtData, vbString
      ' Write data to the file.
      Print #1
   Case icDocEnd
      ' Close the file.
      Close #1
   End Select

End Sub
```

SetData Method (DocOutput Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDocOutputSetDataC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDocOutputSetDataX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDocOutputSetDataA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDocOutputSetDataS"}

Overrides the next data buffer to be transferred when the DocOutput event is activated.

Return Value

Void

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**SetData** *data*
Visual FoxPro	*Object*.SetData(*data)*

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

Arguments

Development Tool	Argument	Description
Microsoft Access and Visual Basic	*data*	Next block of data to be sent. For binary data, byte array should be used.
Visual FoxPro	*data*	Next block of data to be sent. For binary data, byte array should be used.

Remarks

SetData can be called during DocOutput event handling (when the **State** property is set to **icDocData**) to change the next buffer of data to be transferred. Calling **SetData** modifies the data received by any **DocInput** objects linked to this **DocOutput** object using data linking, as well as the data written to the output file if one is specified via the **FileName** property.

PushStreamMode Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthPushStreamModeMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthPushStreamModeMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthPushStreamModeMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthPushStreamModeMethodS"}

Performs the next step of the document transfer. This method is called when you implement data streaming.

Return Value

Void

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object.***PushStream**
Visual FoxPro	*Object.*PushStream

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

Arguments

None.

Remarks

User implementation of data streaming is handled by the **PushStreamMode** property and the **PushStream** method. These interfaces are only important if you implement data streaming.

Input data streaming can be implemented in two ways:

The **PushStreamMode** property is set to False (the default), and data is specified when the DocInput event is activated. You should not call **PushStream**.

PushStreamMode is set to True, and when data is available call the **PushStream** method. **PushStream** is called to perform the next step of the document transfer. **PushStream** changes the **State** property based on the next step of the transfer, activates the DocInput event as needed, and returns to wait for the next call to **PushStream**.

When using this technique, instead of setting document information in the DocInput event handler, set document information before calling **PushStream**. For example, when transferring data, invoke the **SetData** method before calling **PushStream**.

When using an input file (**FileName** property) or data linking (**DocLink** property), the control calls the **PushStream** method. In these cases you can not call it.

BytesTotal Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBytesTotalPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBytesTotalPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBytesTotalPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBytesTotalPropertyS"}

Returns the total bytes to be transferred or zero, if not available. Read-only and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**BytesTotal**
Visual FoxPro	*Object*.BytesTotal

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

Return Values

Development Tool	Default Value	Data Type
Microsoft Access and Visual Basic	Zero	Long
Visual FoxPro	0	Numeric

Remarks

The BytesTotal property is available as soon as document transfer begins. The property value does not change until a new transfer begins. This value is/can be zero if the size of the document is unknown.

BytesTransferred Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBytesTransferredPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBytesTransferredPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBytesTransferredPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBytesTransferredPropertyS"}

Returns the number of bytes transferred. Read-only and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object.***BytesTransferred**
Visual FoxPro	*Object*.BytesTransferred

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

Return Values

Development Tool	Default Value	Data Type
Microsoft Access and Visual Basic	Empty string	Long
Visual FoxPro	0	Numeric

Remarks

The **BytesTransferred** property is updated as document transfer progresses. This property value is set to zero when a new transfer begins, and it is updated before the DocInput or DocOuput event occurs. The value is not changed after the transfer is complete (it will reflect the total for the last transfer when no transfer is in progress).

SetData Method (DocInput Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthSetDataMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthSetDataMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthSetDataMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthSetDataMethodS"}

Specifies the next data buffer to be transferred when the DocInput event occurs.

Return Value

Void

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**SetData** *Data*
Visual FoxPro	*Object*.SetData(*Data*)

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

Arguments

Development Tool	Argument	Description
Microsoft Access and Visual Basic	*Data*	Next block of data to be sent. For binary data, byte array should be used.
Visual FoxPro	*Data*	Next block of data to be sent. For binary data, byte array should be used.

Remarks

SetData is generally invoked during DocInput event handling (when the **State** property is set to icDocData) to specify the next buffer of data to be transferred. **SetData** can also be invoked before **SendDoc** to specify the initial buffer of data to be transferred. This is an alternative to passing the InputData argument to **SendDoc**. If you implement data streaming using **PushStreamMode**, you can also invoke **SetData** before **PushStream**.

When using an input file (**FileName** property) or input link (**DocLink** property), **SetData** can be invoked during DocInput event handling to change the next buffer of data to be transferred. Calling **SetData** in these cases modifies the data transferred to the target document.

Suspend Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthSuspendMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthSuspendMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthSuspendMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthSuspendMethodS"}

Suspends or resumes document transfer.

Return Value

Void

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**Suspend** *suspend*
Visual FoxPro	*Object*.Suspend(*ISuspend*)

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

Arguments

Development Tool	Argument	Description
Microsoft Access and Visual Basic	*suspend*	Indicates whether to suspend or resume transfer. See Settings below.
Visual FoxPro	*ISuspend*	Indicates whether to suspend or resume transfer. See Settings below.

Settings

The settings for **Suspend** are:

Argument	Description
True	The transfer is suspended.
False	The transfer is resumed.

Remarks

Calls to Suspend with True and False arguments must be balanced. For example, if Suspend(True) is called twice, Suspend(False) must be called twice to resume transfer.

PushStreamMode Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproPushStreamModePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPushStreamModePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPushStreamModePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPushStreamModePropertyS"}

Returns or sets a value which indicates whether the stream is in push or pull mode.

- For the **DocOutput** object, this property is read-only and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**PushStreamMode** [= *boolean*]
Visual FoxPro	*Object*.PushStreamMode[= *lExpression*]

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

The **PushStreamMode** property settings are:

Setting	Description
True	The control is in Push mode.
False	The control is in Pull mode.

Data Type

Boolean

Remarks

The **PushStreamMode** of the **DocOutput** object is not needed by the user. The **DocOutput** Object doesn't have a **PushStream** method.

DocInput Object Specific Remarks

User implementation of data streaming is handled by the **PushStreamMode** property and **PushStream** method. These interfaces are only important if you implement data streaming.

Input data streaming can be implemented in two ways:

Set the **PushStreamMode** property to **False** (the default) and data is specified when the DocInput event is activated.

Set the **PushStreamMode** property to **True** before initiating the document transfer. See the **PushStream** method for more information on this technique.

When using an input file (**FileName** property) or data linking (**DocLink** property), the control sets the **PushStreamMode** property. In this case, you can not set it.

DocLink Property (DocInput Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDocLinkPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDocLinkPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDocLinkPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDocLinkPropertyS"}

Returns or sets a copy of a **DocOutput.DocLink** property when data linking is used or empty if data linking is not used. Read/write and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**DocLink** [=*string*]
Visual FoxPro	*Object*.DocLink[= *string*]

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

Return Values

Development Tool	Default Value	Data Type
Microsoft Access and Visual Basic	None	DocLink
Visual FoxPro	None	

Remarks

The **DocLink** property can be set before calling the **SendDoc** method in a particular control. It should be set to the **DocLink** property of a **DocOutput** object. This causes data output from the **DocOutput** object to be used as the data input for the **DocInput** object.

Input data can only be supplied through an input file, data linking, or data streaming. Property contents determine how the data is supplied. If the **FileName** property is not empty, it is used as the input file. If the **DocLink** property is not empty, data linking is used. Otherwise, data streaming via the DocInput event is used.

When the **DocLink** property is set to a nonempty value, the **FileName** property is automatically set to empty.

Headers Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproHeadersPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHeadersPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHeadersPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHeadersPropertyS"}

Returns a reference to a the **DocHeaders** collection. Read-only and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**Headers**
Visual FoxPro	*Object*.Headers

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

Data Type

DocHeaders.

Remarks

The contents of the **DocHeaders** collection can be modified before calling the **GetDoc** or **SendDoc** method of the specific control, or it can be passed as an argument to these methods. If it is passed as an argument, the items in the **DocInput.Headers** collection are replaced with those in the collection specified in the argument.

DocHeader Objects

The **DocHeaders** collection contains **DocHeader** objects, each of which represents a MIME (Multipurpose Internet Mail Extensions) header and contains a **Name** and **Value** property. For example, an item with a **Name** of `content-type` will have a **Value** indicating the document type such as `"text/plain"` or `"image/gif"`. The headers used depends on the protocol, however two headers are common to all protocols: content-type and content-length.

`content-type` indicates the document type as specified by MIME.

`content-length` indicates the size of the document in bytes.

DocOutput Object Specific Remarks

The contents of the **DocHeaders** collection are set during the document transfer to headers that describe information about the output document. When **SendDoc** is called for protocols that always send a reply document, these headers describe information about the reply document.

Authenticate Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtAuthenticateEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtAuthenticateEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtAuthenticateEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtAuthenticateEventS"}

Occurs after an **Authenticate** method is invoked.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*_**Authenticate** ()
Visual FoxPro	PROCEDURE *Object.*Authenticate
Visual C++	**void** *dialogclass*::OnAuthenticate*Control*( );

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

Remarks

Use the **ReplyString** property to determine the server reply after authentication

Authenticate Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthAuthenticateMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthAuthenticateMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthAuthenticateMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthAuthenticateMethodS"}

Authenticates the user based on the parameters passed. If no parameters are passed, the **UserId** and **Password** properties are used. When authentication process is terminated, the Authenticate event is activated.

Return Value

Void

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object.***Authenticate** [*UserID,*] [*Password*]
Visual FoxPro	*Object.*Authenticate([*cUserID*] [, *cPassword*])
Visual C++	**void Authenticate**(**const VARIANT&** *UserId*, **const VARIANT&** *Password*);

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

Arguments

Development Tool	Argument	Data Type	Description
Microsoft Access and Visual Basic	*UserID*	String	Optional. User identification string to use for authentication For Input only.
	Password	String	Optional. Password to use for authentication. For Input only.
Visual FoxPro	*cUserID*	Character expression	Optional. User identification string to use for authentication For Input only.
	cPassword	Character expression	Optional. Password to use for authentication. For Input only.
Visual C++	*UserID*	**VARIANT**	Optional. User identification string to use for authentication For Input only.
	Password	**VARIANT**	Optional. Password to use for authentication.

For Input only.

Remarks

Optional arguments to this method override the values from corresponding **UserId** and **Password** properties. The values of the properties will not change. If you omit any of the arguments, the value from a corresponding property will be used to provide authentication.

Authenticate Method, Authenticate Event Example

The **Authenticate** method can take two arguments which are supplied by the **UserId** and **Password** properties. The example shows a typical use of the **Authenticate** method when logging onto public FTP servers. It's common to use "Anonymous" as the UserId, and the user's email address to log on.

```
Private Sub cmdAuthenticate_Click()
   FTP1.UserId = "Anonymous"
   FTP1.Password = "JohnD@internet.com"
   FTP1.Authenticate ' The method uses the values
                     ' supplied by the UserID and
                     ' Password properties.
End Sub

Private Sub FTP1_Authenticate()
   MsgBox FTP1.ReplyString ' Returns the server's reply.
End Sub
```

Password Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproPasswordPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPasswordPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPasswordPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPasswordPropertyS"}

Returns or sets the password of the current user on the server. Read/Write and available at run time and design time.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**Password** [= *string*]
Visual FoxPro	*Object*.Password[= *cPassword*]
Visual C++	**CString GetPassword**( ); **void SetPassword**(**LPCTSTR** *lpszNewValue*);

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

Data Type

String (**CString** object in Visual C++)

Remarks

If the **UserId** and the **Password** properties are set before invoking the **Authenticate** method, the arguments for the **Authenticate** method need not be specified. This is shown in the code below:

```
Private Sub cmdAuthenticate_Click()
   FTP1.UserID = "anonymous"
   FTP1.Password = "johnD@Mycompany.com"
   FTP1.Authenticate ' UserId and password arguments
                     ' aren't required if set previously.
End Sub
```

Quit Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtQuitEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtQuitEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtQuitEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtQuitEventS"}

Occurs after the **Quit** method is invoked.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*_**Quit**
Visual FoxPro	PROCEDURE *Object.*Quit
Visual C++	**void** *dialogclass*::OnQuit*Control*( );

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

Arguments

None

Busy Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtBusyEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtBusyEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtBusyEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtBusyEventS"}

Occurs when a command is in progress or when a command has completed.

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*_**Busy**(**ByVal** *isBusy* **As Boolean**)
Microsoft Visual FoxPro	PROCEDURE *Object.*Busy LPARAMETERS *lBusy*
Microsoft Visual C++	**void** *dialogclass*::OnBusy*Control*(**BOOL** *isBusy*);

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

Arguments

Development Tool	Argument	Description
Microsoft Access and Visual Basic	*isBusy*	Indicates whether or not a command is in progress.
Visual FoxPro	*lBusy*	Indicates whether or not a command is in progress.
Microsoft Visual C++	*isBusy*	Indicates whether or not a command is in progress.

Settings

The settings for *Busy* are:

Setting	Description
True	A command is in progress.
False	No command is in progress.

Busy Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBusyPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBusyPropertyX"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBusyPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBusyPropertyS"}

Returns a value indicating whether a command is in progress. Read-only and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*.**Busy**
Microsoft Visual FoxPro	*Object*.Busy
Microsoft Visual C++	**BOOL GetBusy**( );

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

Settings

The **Busy** property settings are:

Setting	Description
True	A command is in progress.
False	No command is in progress.

Cancel Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtCancelEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtCancelEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtCancelEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtCancelEventS"}

Occurs after a cancellation request has been completed and satisfied. After this event, the object's state changes to idle.

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*_**Cancel** ()
Microsoft Visual FoxPro	PROCEDURE *Object*.Cancel
Microsoft Visual C++	**void** *dialogClass*::OnCancel*Control*( );

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

Arguments

None

Cancel Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthCancelMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthCancelMethodX"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthCancelMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthCancelMethodS"}

Cancels a pending request.

Return Value

Void

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*.**Cancel**
Microsoft Visual FoxPro	*Object*.Cancel( )
Microsoft Visual C++	**void Cancel**( );

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

Arguments

None

DocInput Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtDocInputEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtDocInputEventX"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtDocInputEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtDocInputEventS"}

Occurs when input data has been transferred from a control.

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*_**DocInput** (**ByVal** *DocInput* **As DocInput**)
Microsoft Visual FoxPro	PROCEDURE *Object*.DocInput LPARAMETERS *oDocInput*

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

Arguments

Development Tool	Argument	Description
Microsoft Access and Visual Basic	*DocInput*	Object describing document input data for the current transfer.
Microsoft Visual FoxPro	*oDocInput*	Object describing document input data for the current transfer.

Remarks

Use the properties and methods of the **DocInput** object to assess the state of a data transfer. For example, you can construct a progress bar using the values of the **BytesTotal**, **BytesTransferred** and **State** properties. This is shown in the code below:

```
Sub HTTP1_DocInput(DocInput As DocInput)
   Select Case DocInput.State
   Case icDocBegin ' Begin transfer
      ProgressBar1.Visible = True ' Show bar.
      ProgressBar1.Max = DocInput.BytesTotal
      ProgressBar1.Value = 0
   Case icDocData ' Reset bar value.
      ProgressBar1.Value = DocInput.BytesTransferred
   Case icDocEnd
      ProgressBar1.Visible = False ' Hide bar.
   End Select
End Sub
```

For more information See "Using DocInput and DocOutput Objects."

DocInput Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDocInputPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDocInputPropertyX"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDocInputPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDocInputPropertyS"}

Returns a reference to a **DocInput** object. Use this property to access properties of the **DocInput** object. Read-only and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*.**DocInput**
Microsoft Visual FoxPro	*Object*.DocInput

Data Type

DocInput

Remarks

Properties of the **DocInput** object can be set before invoking the **SendDoc** method or they can be passed as arguments to this method.

```
NNTP1.DocInput.FileName = "messages.txt"
NNTP1.DocInput.Headers = myDocHeaders
NNTP1.SendDoc URLs
   ' This can also be coded as:
NNTP1.SendDoc URLs, myDocHeaders, "messages.txt"
```

The **DocInput** object is also used for conveying information about the progress of the document transfer and for data linking and streaming. Use the BytesTransferred and BytesTotal properties as shown below:

```
' Assuming there is a label called lblStatus.
lblStatus.Caption = NNTP1.DocInput.BytesTransferred
```

DocOutput Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtDocOutputEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtDocOutputEventX"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtDocOutputEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtDocOutputEventS"}

Occurs when output data has been transferred from the control. Use the reference to the **DocOuput** object contained in this event to parse data and send it to an appropriate destination.

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*_**DocOutput** (**ByVal** *DocOutput* **As DocOutput**)
Microsoft Visual FoxPro	PROCEDURE *Object*.DocOutput LPARAMETERS *oDocOutput*

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

Arguments

Development Tool	Argument	Description
Microsoft Access and Visual Basic	*DocOutput*	Object describing document output data for the current transfer.
Microsoft Visual FoxPro	*DocOutput*	Object describing document output data for the current transfer.

Remarks

The reference to the **DocOutput** object contained in this event can be used to access all the properties and methods of the **DocOutput** object. Use the State property to examine the data stream before processing it further. The code example shows the basic method of using the State property with a Select Case statement.

```
NNTP1_DocOutput(DocOuput As DocOutput)
   Dim vtData As Variant ' Data variable.
   Select Case DocOutput.State
   Case icDocNone '
      ' Handle no event here
   Case icDocBegin
      ' Handle beginning of doc transfer.
   Case icData ' Data
      ' Retrieve the Data
      DocOutput.GetData vtData
      txtData = txtData & vbCrLf & vtData
   Case icDocError
      ' Handle errors here.
   Case DocHeaders
      txtHeads = txtHeads & vbCrLf & DocOuput.Headers
   Case icDocEnd
      ' Handle end of transfer here.
   End Select
End Sub
```

For more information See "Using DocInput and DocOutput Objects."

DocOutput Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDocOutputPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDocOutputPropertyX"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDocOutputPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDocOutputPropertyS"}

Returns a reference to the **DocOutput** object. Returns a reference to a **DocInput** object. Use this property to access properties of the **DocOutput** object. Read-only and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*.**DocOutput**
Microsoft Visual FoxPro	*Object*.DocOutput

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

Data Type

DocOutput

Remarks

Properties of the **DocOutput** object can be set before invoking the **GetDoc** method or they can be passed as arguments to this method.

```
NNTP1.DocOutput.FileName = "messages.txt"
NNTP1.DocOutput.Headers = myDocHeaders
NNTP1.GetDoc URLs
   ' This can also be coded as:
NNTP1. GetDoc URLs, myDocHeaders, "messages.txt"
```

The **DocOutput** object is also used for conveying information about the progress of the document transfer and for data linking and streaming. Use the **BytesTransferred** and **BytesTotal** properties as shown below:

```
' Assuming there is a label called lblStatus.
lblStatus.Caption = NNTP1. DocOutput.BytesTransferred
```

EnableTimer Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproEnableTimerPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproEnableTimerPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproEnableTimerPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproEnableTimerPropertyS"}

Sets a value that enables the TimeOut event. Write only at run time.

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*.**EnableTimer(***event***)** [= *boolean*]
Microsoft Visual FoxPro	*Object*.EnableTimer(*nEvent*) [= *lExpression*]
Microsoft Visual C++	**void SetEnableTimer**(**short** *event*, **BOOL** *bNewValue*);

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

Arguments

Development Tool	Argument	Data Type	Description
Microsoft Access and Visual Basic	*event*	Integer	A value that determines the type of Timeout event that will be enabled. See Settings below.
	boolean	Boolean	A value that determines if the specified TimeOut event will be enabled. See Settings below.
Visual FoxPro	*nEvent*	Numeric	A value that determines the type of Timeout event that will be enabled. See Settings below.
	lExpression	Logical	A value that determines if the specified TimeOut event will be enabled. See Settings below.
Visual C++	*event*	Integer	A value that determines the type of Timeout event that will be enabled. See Settings below.
	bNewValue	Boolean	A value that determines the type of Timeout event that will be enabled. See Settings below.

Settings

The settings for *event* and *nevent* are:

Constant	Value	Description
prcConnectTimeout	1	Timeout for connect. If connection is not established within the timeout period, Timeout event will be fired.
prcReceiveTimeout	2	Timeout for receiving data. If no data arrives within the timeout period, Timeout event will be fired.

prcUserTimeout	65	Timeout for user defined event. User should use prcUserTimeout + [Integer] range for custom timeout events.

The settings for *boolean* and *lExpression* are:

Constant	Value	Description
True	-1	The timer for this event will be enabled.
False	0	The timer for this event will not be enabled.

Remarks

Use the **TimeOut** property to set the time for any event.

You can enable the timer for each of the different event types using the code below:

```
With NNTP1
   .EnableTimer(prcConnectTimeout)= True
   .EnableTimer(prcReceiveTimeout)= True
   .EnableTimer(prcUserTimeout)= True
End With
```

Errors Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproErrorsPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproErrorsPropertyX"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproErrorsPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproErrorsPropertyS"}

Returns a reference to the **icErrors** collection. The collection can be accessed for details about the last error that occurred. Read-only and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*.**Errors**
Microsoft Visual FoxPro	*Object*.Errors

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

Data Type

Errors

Remarks

The **Errors** property can be used within an Error event if information passed through the Error event is not sufficient, as shown below.

```
Private Sub HTTPCT1_Error(Number As Integer, _
Description As String, Scode As Long, Source As _
String, HelpFile As String, HelpContext As _
Long, CancelDisplay As Boolean)
   Dim strErrs As String   ' String variable.
   Dim errCol As icErrors  ' icErrors collection
                           ' variable.
   Set errCol = HTTP1.Errors ' Set the variable.
   ' Iterate through the collection and place
   ' results in a string variable.
   For Each icError in errCol
      strErrs = strErrs & icError.Code & _
      icError.Description & " " & vbCrLf
   Next

   MsgBox strErrs ' Show all errors.
End Sub
```

icError Object, icErrors Collection, Errors Property Example

The example below uses the **Errors** Property of an **HTTP** control to retrieve a reference to the **icErrors** collection. The code executes when the DocOutput event occurs, which contains a reference to the **DocOutput** object. If the **State** property of the **DocOutput** object is **icDocError** (4), then the code iterates through each **icError** object and places the **Code** and **Description** into a string variable. Finally, the code writes the string variable into a **TextBox** control named "txtErrors."

```
Sub HTTPCT1.DocOutput(ByVal DocOutput As DocOutput)
   Dim strErrs As String   ' Message variable.
   Dim errCol As icErrors  ' icErrors collection
                           ' variable.

   Select Case DocOutput.State
   Case icDocError
      Set errCol = HTTP1.Errors ' Set the variable.

      ' Iterate through the collection and place
      ' results in a string variable.
      For Each icError in errCol
         strErrs = strErrs & icError.Code & _
         icError.Description & " " & vbCrLf
      Next
   Case icDocData
      ' Handle DocData.
   End Select

   ' Write the collection into a TextBox control named
   ' txtErrors.
   txtErrors.Text = txtErrors & strErrs
End Sub
```

GetDoc Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthGetDocMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthGetDocMethodX"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthGetDocMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthGetDocMethodS"}

A **DocOutput** related method that requests retrieval of a document identified by a URL.

Return Value

Void

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*.**GetDoc** [*URL*,] [*Headers*,] [*OutputFile*]
Microsoft Visual FoxPro	*Object.*GetDoc([c*URL*] [, *cHeaders*] [, *cOutputFile*])
Microsoft Visual C++	**void GetDoc**(**const VARIANT&** *URL*, **const VARIANT&** *Headers*, **const VARIANT&** *OutputFile*);

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

Arguments

Microsoft Access and Visual Basic

Argument	Default	Data Type	Description
URL	DocInput. URL	String	Optional. The URL identifying the remote document to be retrieved. For Input only.
Headers	DocInput. Headers	DocHeaders	Optional. Headers used for requesting the document. This argument only applies to protocols where request headers can be specified (for example, HTTP). For Input only.
OutputFile	DocOutput. Filename	String	Optional. A local file to which the retrieved document will be written.

Microsoft Visual FoxPro

Argument	Default	Data Type	Description
cURL	DocInput.URL	Character expression	Optional. The URL identifying the remote document to be retrieved. For Input only.

cHeaders	DocInput.Headers	Character expression	Optional. Headers used for requesting the document. This argument only applies to protocols where request headers can be specified (for example, HTTP). For Input only.
cOutputFile	DocOutput.Filename	Character expression	Optional. A local file to which the retrieved document will be written.

In Microsoft Visual C++, all the parameter types of **GetDoc** are **VARIANT**.

Remarks

The **GetDoc** method permits retrieving a file from the server.

The URL and (for some controls) Headers are used as inputs specifying which document is to be retrieved. The OutputFile argument indicates where the retrieved document should be written locally.

The URL type (first part up to the colon) can be omitted and defaults to the correct type for this control. For example, when using the **HTTP** control, the "http:" string can be omitted.

For basic use of this control, arguments should be passed to **GetDoc** to describe the document transfer. For more powerful use of this control, the **DocInput** and **DocOutput** objects can be used in conjunction with the DocInput and DocOutput events. The arguments of **GetDoc** correspond to properties in the **DocInput** and **DocOutput** objects of this control. **DocInput** and **DocOutput** properties can be set before calling **GetDoc** to avoid passing arguments. The DocInput and DocOutput events can also be used for transferring data using streaming rather than local files.

Method Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproMethodPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMethodPropertyX"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMethodPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMethodPropertyS"}

Returns or sets the method used to retrieve or post (send) the document. Available at run time and design time.

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*.**Method** [= *integer*]
Microsoft Visual FoxPro	*Object*.Method[= *nValue*]
Microsoft Visual C++	**long GetMethod**( ); **void SetMethod**(**long** *nNewValue*);

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

Data Type

Integer.

Settings

The possible values for *integer* and *nValue* are:

Constant	Value	Description
prcGet	1	Default. **Get** method requests the whole document.
PrcHead	2	**Head** method requests only the headers of a document.
PrcPost	3	**Post** method posts a document to the server as a sub-ordinate of the document specified by the URL.
PrcPut	4	**Put** method method puts a document to the server. The document is to replace an existing document specified by the URL.

ProtocolStateChanged Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtProtocolStateChangedEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtProtocolStateChangedEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtProtocolStateChangedEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtProtocolStateChangedEventS"}

This event occurs whenever the protocol state changes.

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*_**ProtocolStateChanged** (**ByVal** *ProtocolState* **As Integer**)
Microsoft Visual FoxPro	PARAMETERS *Object*.ProtocolStateChanged LPARAMETERS n*State*
Microsoft Visual C++	**void** *dialogclass*::OnProtocolStateChanged*Control*(**short** *ProtocolState*);

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

Arguments

Refer to the **ProctocolState** property for possible values of the state argument.

SendDoc Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthSendDocMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthSendDocMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthSendDocMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthSendDocMethodS"}

A **DocInput** related method that requests sending a document identified by a URL.

Return Value

Void

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*.**SendDoc** [*URL*,] [*Headers*,] [*InputData*,] [*InputFile*,] [*OutputFile*]
Microsoft Visual FoxPro	*Object*.SendDoc([*cURL*] [, *cHeaders*] [, *eInputData*] [, *cInputFileName*] [, *cOutputFileName*])
Microsoft Visual C++	**void SendDoc**(**const VARIANT&** *URL*, **const VARIANT&** *Headers*, **const VARIANT&** *InputData*, **const VARIANT&** *InputFile*, **const VARIANT&** *OutputFile*);

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

Arguments

Microsoft Access and Visual Basic

Argument	Default	Data Type	Description
URL	DocInput.URL	STRING	Optional. The URL identifying the remote document to be sent. If specified, the URL property will be set to this value. For input only.
Headers	DocInput. Headers	DocHeaders	Optional. Headers used for sending the document. This argument only applies to protocols where document headers can be sent (for example, SMTP and HTTP). For input only.
InputData	DocInput. SetData	Variant	Optional. A data buffer containing the document to be sent. For input only.
InputFile	DocInput. Filename	STRING	Optional. A local file containing the document to be sent. For input only.
OutputFile	DocOutput. Filename	STRING	Optional. A local file to which a reply document is written. This argument only applies for protocols that return a

			reply document (for example, HTTP). For input only.

Visual FoxPro

Argument	Default	Data Type	Description
cURL	DocInput.URL	Character expression	Optional. The URL identifying the remote document to be sent. If specified, the URL property will be set to this value. For input only.
cHeaders	DocInput. Headers	Character expression	Optional. Headers used for sending the document. This argument only applies to protocols where document headers can be sent (for example, SMTP and HTTP). For input only.
eInputData	DocInput. SetData		Optional. A data buffer containing the document to be sent. For input only.
cInputFile	DocInput. Filename	Character expression	Optional. A local file containing the document to be sent. For input only.
cOutputFile	DocOutput. Filename	Character expression	Optional. A local file to which a reply document is written. This argument only applies for protocols that return a reply document (for example, HTTP). For input only.

In Microsoft Visual C++, all the parameter types of **SendDoc** are **VARIANT**.

Remarks

The **SendDoc** method permits sending (posting or putting) a file to the server.

The URL and (for some controls) Headers are used as inputs describing the document to be sent. The InputData and InputFile arguments (only one can be specified) contain the document to be sent. For controls such as **HTTP** that return a reply document, the OutputFile argument indicates where the reply document should be written locally.

The URL type (first part up to the colon) can be omitted and defaults to the correct type for this control. For example, when using the **HTTP** control, the "http:" string can be omitted.

For basic use of this control, arguments should be passed to **SendDoc** to describe the document transfer. For more powerful use of this control, the **DocInput** and **DocOutput** objects can be used in conjunction with the DocInput and DocOutput events. The arguments of **SendDoc** correspond to properties in the **DocInput** and **DocOutput** objects of this control. **DocInput** and **DocOutput** properties can be set before calling **SendDoc** to avoid passing arguments. The DocInput and DocOutput events can also be used for transferring data using streaming rather than local files.

State Property (Internet Control Pack)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproICPStateC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproICPStateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproICPStateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproICPStateS"}

Returns the connection state of the control. Read-only and unvailable at design time.

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*.**State**
Microsoft Visual FoxPro	*Object*.State
Microsoft Visual C++	**short GetState**( );

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

Data Type

Microsoft Access, Visual Basic, and Visual C++	Visual FoxPro
Integer	Numeric

Settings

The settings for the **State** property are:

Constant	Value	Description
prcConnecting	1	Connecting. Connect has been requested, waiting for connect acknowledge.
prcResolvingHost	2	Resolving Host. This state happens only if **RemoteHost** property is in name format (rather than dot-delimited IP format)
prcHostResolved	3	Resolved the host. This state occurs only if ResolvingHost state has been entered previously
prcConnected	4	Connection established.
prcDisconnecting	5	Connection close/disconnect has been initiated
prcDisconnected	6	This is the initial state when the protocol object is instantiated, before Connect has been been initiated or after a Connect attempt failed or after Disconnect was performed.

StateChanged Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtStateChangedEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtStateChangedEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtStateChangedEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtStateChangedEventS"}

Occurs whenever the transport state changes. The state is given in the **State** property.

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*_**StateChanged** (**ByVal** *State* **As Integer**)
Microsoft Visual FoxPro	PROCEDURE *Object*.StateChanged LPARAMETERS *nState*
Microsoft Visual C++	**void** *dialogclass*::OnStateChanged*Control*(**short** *State*);

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

Settings

The settings for the *State* and *nState* are:

Constant	Value	Description
prcConnecting	1	Connecting. Connect has been requested, waiting for connect acknowledge.
prcResolvingHost	2	Resolving Host. This state happens only if **RemoteHost** property is in name format (rather than dot-delimited IP format)
prcHostResolved	3	Resolved the host. This state occurs only if ResolvingHost state has been entered previously
prcConnected	4	Connection established.
prcDisconnecting	5	Connection close/disconnect has been initiated
prcDisconnected	6	This is the initial state when the protocol object is instantiated, before Connect has been been initiated or after a Connect attempt failed or after Disconnect was performed.

TimeOut Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtTimeOutEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtTimeOutEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtTimeOutEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtTimeOutEventS"}

Occurs when the specified event does not take place within the interval defined in the property **Timeout** for that event. Set *continue* to **True** to continue.

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*_**TimeOut** (**ByVal** *event* **As Integer**, *Continue* **As Boolean**)
Microsoft Visual FoxPro	PROCEDURE *Object.*TimeOut LPARAMETERS *nEvent*, *lContinue*
Microsoft Visual C++	**void** *dialogclass*::OnTimeout*Control*(**short** *event*, **BOOL FAR*** *Continue*);

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

Arguments

Development Tool	Argument	Description
Microsoft Access and Visual Basic	*event*	Defines the event to which the time interval applies.
	Continue	Boolean value that determines if the timer will remain active.
Visual FoxPro	*nEvent*	Defines the event to which the time interval applies.
	lContinue	Logical value that determines if the timer will remain active.Defines the event to which the time interval applies.

Settings

The settings for *event* and *nEvent* are:

Constant	Value	Description
prcConnectTimeout	1	Timeout for connect. If connection is not established within the timeout period, Timeout event will be fired.
prcReceiveTimeout	2	Timeout for receiving data. If no data arrives within the timeout period, Timeout event will be fired.
prcUserTimeout	65	Timeout for user defined event. User should use prcUserTimeout + [Integer] range for custom timeout events.

The settings for *continue* are:

Setting	Description
True	The timer continues.
False	(Default) The timer stops.

Remarks

Use the event argument to determine the type of event that has expired. For example, the following code handles all three cases:

```
HTML_Timeout(ByVal event as Integer, continue As Boolean)
Select Case event
   Case prcConnectTimeout
      ' Handle the connection timeout.
   Case prcReceiveTimeout
      ' Ignore the Receive timeout.
   Case prcUserTimeout
      ' Handle the User defined timeout.
End Select
End Sub
```

URL Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproURLPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproURLPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproURLPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproURLPropertyS"}

Returns or sets a Universal Resource Locator (URL) string identifying the current document being transferred. Read/write and available at run time.

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*.**URL** [= S*tring*]
Microsoft Visual FoxPro	*Object*.URL[= *cURLString*]
Microsoft Visual C++	**CString GetURL**( ); **void SetURL**(**LPCTSTR** *lpszNewValue*);

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

Arguments

Development Tool	Argument	Description
Microsoft Access and Visual Basic	S*tring*	Valid URL.
Visual FoxPro	*cURLString*	Valid URL.
Visual C++	*lpszNewValue*	Valid URL.

Data Type

String (**CString** in Visual C++)

Remarks

URL may be set before calling the **GetDoc** or **SendDoc** method of the control, or it may be passed as an argument to these methods. If it is passed as an argument, the **URL** property will be set to the argument value.

In the **HTTP** control, the **URL** property identifies an HTTP request of any kind. The URL type (first part up to the colon) may be omitted. In this case, it will default to the correct type for this control. For example, the `http`: string may be omitted when using the **HTTP** control.

UserID Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproUserIdPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproUserIdPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproUserIdPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproUserIdPropertyS"}

Returns or sets the user identification number or name to be used during authentication transactions. Read/Write and available at run time and design time.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**UserId** [= *string*]
Visual FoxPro	*Object*.UserId[= *cIDNumber*]
Visual C++	**CString GetUserId**( ); **void SetUserId**(**LPCTSTR** *lpszNewValue*);

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

Data Type

String (**CString** object in Visual C++)

Remarks

If the **UserId** and the **Password** properties are set before invoking the **Authenticate** method, the arguments for the **Authenticate** method need not be specified. This is shown in the code below:

```
Private Sub cmdAuthenticate_Click()
   FTP1.UserID = "anonymous"
   FTP1.Password = "johnD@Mycompany.com"
   FTP1.Authenticate ' UserId and password arguments
                     ' aren't required if set previously.
End Sub
```

Connect Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthConnectMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthConnectMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthConnectMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthConnectMethodS"}

Initiates connection to a remote machine. If a connection is established, the StateChanged event occurs.

Return Value

void.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**Connect** [*RemoteHost*,] [*RemotePort*]
Visual FoxPro	*Object*.Connect([*cRemoteHost*] [, *nRemotePort*])
Visual C++	**void Connect**(**const Variant&** *RemoteHost*, **const Variant&** *RemotePort*);

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

Arguments

Development Tool	Argument	Data Type	Description
Microsoft Access, Visual Basic, and Visual C++	*RemoteHost*	Variant	Optional. If this argument is missing, **Connect** will connect to the remote host specified in the **RemoteHost** property. For Input only.
	RemotePort	Variant	Optional. If this parameter is missing, **Connect** will connect to the remote port specified in the **RemotePort** property. For Input only.
Visual FoxPro	*cRemoteHost*	Character	Optional. If this parameter is missing, **Connect** will connect to the remote host specified in the **RemoteHost** property. For Input only.
	nRemotePort	Numeric	Optional. If this parameter is missing, **Connect** will connect to the remote port specified in the **RemotePort** property. For Input only.

Remarks

If the connection is successfully established, the Connect event will occur. If an error occurs during connection, the Error event will occur

Error Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtErrorEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtErrorEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtErrorEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtErrorEventS"}

Occurs whenever an error occurs in background processing (for example, failed to connect, or failed to send or receive in the background).

Syntax

Development Tool	Syntax
Access and Visual Basic	*object*_**Error(***ErrCode* **As Integer,** *Description* **As String,** *Scode* **As Long,** *Source* **As String,** *HelpFile* **As String,** *HelpContext* **As Long,** *CancelDisplay* **As Boolean)**
Visual FoxPro	PROCEDURE *Object*.Error LPARAMETERS *nShortErrCode*, *cDescription*, *nLongErrorCode*, *cSource*, *cHelpFile*, *nHelpContextID*, *lCancelDisplay*
Visual C++	**void** *dialogclass*::OnError*Control*(**short** *Number*, **BSTR FAR*** *Description*, **long** *Scode*, **LPCTSTR** *Source*, **LPCTSTR** *HelpFile*, **long** *HelpContext*, **BOOL FAR*** *CancelDisplay*);

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

Arguments

Development Tool	Argument	Description
Access and Visual Basic	*ErrCode*	An integer that defines the error code.
	Description	String containing error information.
	Scode	The long SCODE
	Source	String describing the error source.
	HelpFile	String containing the help file name.
	HelpContext	Help file context.
	CancelDisplay	Indicates whether to cancel the display. The default is FALSE, which is to display the default error message box. If you do not want to use the default message box, set CancelDisplay to TRUE.
Visual FoxPro	*nShortErrCode*	An numeric value that defines the error code.
	cDescription	A character string containing error information.
	nLongErrorCode	The numeric long SCODE.
	cSource	A character string describing the error source.
	cHelpFile	A character string containing the help file name.

	nHelpContextID	The numeric Help file context ID.
	lCancelDisplay	Indicates whether to cancel the display. The default is false (.F.), which displays the default error message box. If you do not want to use the default message box, set lCancelDisplay to true (.T.).
Visual C++	*Number*	A **short** that defines the error code.
	Description	32-bit character pointer to a string containing error information.
	Scode	The long SCODE
	Source	32-bit pointer to a constant character string describing the error source.
	HelpFile	32-bit pointer to a constant character string containing the help file name.
	HelpContext	Help file context.
	CancelDisplay	Indicates whether to cancel the display. The default is FALSE, which is to display the default error message box. If you do not want to use the default message box, set CancelDisplay to TRUE.

Internet References

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

The RFC documents cited in the ActiveX™ Controls Pack are available for anonymous downloading from the following site:

`ftp://ds.internic.net/rfc`

Documents available at this site include:

Document	Title	Control
rfc977	Network News Transfer Protocol	NNTP
rfc959	File Tansfer Protocol (FTP)	FTP
rfc821	Simple Mail Transport Protocol	SMTP
rfc1081	Post Office Protocol—Version 3	POP

The MIME (Multipurpose Internet Mail Extensions) protocol (RFC1521) is available at:

```
ftp://thumper.bellcore.com/pub/nsb
http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs
http://www.oac.uci.edu/indiv/ehood/MIME/MIME.html
```

General references on HTML can be found at the following:

```
http://akebono.stanford.edu/yahoo/Computers/World_Wide_Web/HTML/
http://oneworld.wa.com/htmldev.devpage/dev-page.html
```

ProtocolState Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproProtocolStatePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproProtocolStatePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproProtocolStatePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproProtocolStatePropertyS"}

Returns the current state of the protocol. Protocol states vary according to the control. Read-only and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**ProtocolState** [= *integer*]
Visual FoxPro	*Object*.ProtocolState[= *nProtocolState*]
Visual C++	**short GetProtocolState**( );

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

Settings

The settings for *integer*, *nProtocolState* , and the return from **GetProtocolState** are listed by control below:

FTP Control

Constant	Value	Description
ftpBase	0	Default. Base state before connection server is established.
ftpAuthorization	1	Authorization is being performed.
ftpTransaction	2	Authorization has been performed successfully, the client has successfully identified itself to the FTP server.

HTTP Control

Constant	Value	Description
prcBase	0	Default. Base state before connection server is established.
prcTransaction	1	Connection to server is established. This is the valid state for invoking methods with the control.

POP Control

Constant	Value	Description
prcNone	0	Base state before connection server is established.
prcAuthorization	1	Authorization is being performed.
prcTransaction	2	Authorization has been performed successfully, the client has successfully identified itself to the POP3 server and the POP3 server has locked and released the appropriate maildrop.
prcUpdate	3	Occurs when the Quit command is issued from the transaction state.

RemoteHost Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproRemoteHostPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRemoteHostPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRemoteHostPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRemoteHostPropertyS"}

Returns or sets the remote machine to which a control sends or receives data. You can either provide a host name, such as "ftp.microsoft.com," or an IP address string in dotted format, for example "100.0.1.1". If the control uses the **Connect** method, the **RemoteHost** property will be used if the RemoteHost argument is not supplied. Read/Write and available at design time.

- For the Winsock UDP control, returns or sets the remote machine to which to send UDP data.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**RemoteHost** = *string*
Visual FoxPro	*Object*.RemoteHost[= *cRemoteMachine*]
Visual C++	**CString GetRemoteHost**( );

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

Data Type

String (**CString** in Visual C++)

Default Value

Empty

RemotePort Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproRemotePortPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRemotePortPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRemotePortPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRemotePortPropertyS"}

Returns or sets the remote port number to connect to. Read/Write and available at design time.

- For WinSock client applications, this is the remote port number to which to connect if the RemotePort argument of the **Connect** method is not specified.
- For WinSock server applications, after an incoming connection request triggers the ConnectionRequest event, this property contains the port that the remote machine uses to connect to this server.
- For the WinSocket **UDP** control, after the DataArrival event, this property contains the remote port that is sending the UDP data.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**RemotePort** = *long*
Visual FoxPro	*Object*.RemotePort[= *nPortNumber*]
Visual C++	**long GetRemotePort**( ); **void SetRemotePort**(**long** *nNewValue*);

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

Data Type

Development Tool	Type
Microsoft Access, Visual Basic, and Visual C++	Long
Visual FoxPro	Numeric

ReplyCode Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproReplyCodePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproReplyCodePropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproReplyCodePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproReplyCodePropertyS"}

Returns the value of the reply code, which is a protocol-specific number that determines the result of the last request, as returned in the **ReplyString** property. Read-only and unavailable at design time.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**ReplyCode** [= *long*]
Visual FoxPro	*Object*.ReplyCode[= *nReplyValue*]
Visual C++	**long GetReplyCode**( );

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

Data Type

Long

ReplyString Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproReplyStringPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproReplyStringPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproReplyStringPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproReplyStringPropertyS"}

Returns the last reply string sent by the FTP Server to the client as a result of a request.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**ReplyString** [= *String*]
Visual FoxPro	*Object*.ReplyString[= *cReplyString*]
Visual C++	**CString GetReplyString**( );

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

Data Type

String (**CString** in Visual C++)

Remarks

Use the **ReplyString** property in an event to retrieve the result of a request. For example, the following code invokes the **System** method on a **FTP** control. Consequently, the System event occurs, and the server's reply is returned using the **ReplyString** property:

```
Private Sub cmdSystem_Click()
   FTP1.System ' Invoke the System method.
End Sub

Private Sub NNTP1_System()
   ' In the corresponding event, use the ReplyString
   ' property to retrieve the server's reply.
   MsgBox FTP1.ReplyString
End Sub
```

TimeOut Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproTimeOutPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTimeOutPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTimeOutPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTimeOutPropertyS"}

Returns or sets the time that must elapse before the Timeout event occurs.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**TimeOut** [=*long*]
Visual FoxPro	*Object*.TimeOut[= *nTimeoutValue*]
Visual C++	**long GetTimeOut**(**short** *event*); **void SetTimeOut**(**short** *event*, **long** *nNewValue*);

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

Data Type

Long

NotificationMode Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproNotificationModePropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproNotificationModePropertyX"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproNotificationModePropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproNotificationModePropertyS"}

Returns or sets a value that determines when notification is issued for incoming data. Notification can also be suspended.

Syntax

Development Tool	Syntax
Microsoft Visual Basic and Microsoft Access	*object*.**NotificationMode** [= *Integer*]
Microsoft Visual FoxPro	*Object*.NotificationMode[= *nMode*]
Microsoft Visual C++	**short GetNotificationMode**( ); **void SetNotificationMode**(**short** *nNewValue*);

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

Data Type

Integer

Settings

The values for *Integer*, *nMode*, and *nNewValue* are:

Value	Description
0	Default. COMPLETE: notification is provided when there is a complete response.
1	CONTINUOUS: an event is repeatedly activated when new data arrives from the connection.

Connect Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtConnectEventC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtConnectEventX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtConnectEventA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtConnectEventS"}

Occurs when a connection has been successfully established. After this event is triggered, you can send or receive data on the control.

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*_**Connect**
Visual FoxPro	PROCEDURE *Object*.Connect
Visual C++	**void** *dialogclass*::OnConnect*Control*( );

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

Arguments

None

Quit Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthQuitMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthQuitMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthQuitMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthQuitMethodS"}

Initiates a Quit request. If successful, the corresponding Quit event occurs, otherwise the Error event occurs.

Return Value

Void

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**Quit**
Visual FoxPro	*Object.*Quit( )
Visual C++	**void Quit**( );

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

Arguments

None.

NOOP Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthNOOPMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthNOOPMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthNOOPMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthNOOPMethodS"}

Issues the NOOP command to the server. This command requests an OK reply from the server.

Return Value

Void

Syntax

Development Tool	Syntax
Microsoft Access and Visual Basic	*object*.**NOOP**
Visual FoxPro	*Object.*NOOP( )
Visual C++	**void NOOP**( );

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

Arguments

None

Remarks

Use the **ReplyString** property to determine the result of this call.

Getting Started

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

Welcome to the Internet ActiveX™ Controls Pack . The present package includes controls that can be used with Microsoft Access, Visual Basic, Visual C++ and Visual FoxPro. With these controls, you can create applications that work over any network supporting one of the following protocols: TCP/IP, FTP, HTTP, SMTP, and POP3.

Before the introduction of the Internet ActiveX Controls ,, creating applications that could access the rich information sources of the Internet required learning about various low level Winsock Application Programming Interfaces (APIs) as well as the various protocols. The Internet ActiveX Controls hide the complexity of creating Internet programs while exposing the versatility and power of ActiveX controls.

The Internet ActiveX Controlsconsist of the following controls:

- **WinSock TCP control**—used to stream data between networked machines. This is a connection-based control, meaning the communicating machines have an explicit connection.
- **WinSock UDP control**—used to stream data between machines, but without requiring an explicit connection.
- **FTP Client control**—allows you to connect to a remote machine, examine its directories, send and retrieve data using the File Transfer Protocol.
- **HTTP control**—the HyperText Transfer Protocol is the protocol used for World Wide Web sites. This control allows you to gather data which can be parsed, filtered, or placed in a database.
- **HTML control**—The HyperText Markup Language is used to create World Wide Web documents. This control translates the language into viewable pages, allowing you to create your own custom Web browser. The control currently supports HTML version 2.0.
- **POP control**—The Post Office Protocol (version 3) is the most commonly used protocol for email on the Internet. With this control you can create a virtual post office to store email messages.
- **SMTP control**—The Simple Mail Transfer Protocol is used to send email messages on the Internet. With this control you can send email messages from any application.
- **NNTP Client control**— The Network News Transfer Protocol control is used to connect to a news server, retrieve a list of available newsgroups and their descriptions, enter a newsgroup, retrieve lists of articles or particular articles.

Other Requirements for the Internet

To use these controls on the Internet, you also need the following:

- An Internet provider—a local commercial service that provides you access to the Internet.
- A modem or ISDN line—a physical connection between you and your Internet provider.
- Either a PPP (Point to Point Protocol) or a SLIP (Serial Line Internet Protocol)—provided by your Internet service. This allows you to fully access the graphic capabilities of the Internet.

The ActiveX Controls are designed with productivity and ease of use in mind. To this end, all you need – besides the controls – is to connect to your Internet provider, start up your application (Visual Basic, Visual C++, Visual FoxPro, or Microsoft Access), and begin programming.

Requirements for Networks: Know Your Local Protocols

If you are using the controls on a local area network, you need only know which protocols are supported on the network. For example, most networks support the TCP/IP protocol, and this allows you to use the WinSock UDP or TCP controls. The following table shows the protocols supported by the controls.

Control	Protocol Supported
FTP	File Transport Protocol (FTP)

HTML	HyperText Transfer Protocol (HTTP)
HTTP	HTTP
NNTP	Network News Transport Protocol (NNTP)
POP	Post Office Protocol, version 3 (POP3)
SMTP	Simple Mail Transport Protocol (SMTP)
WinSock TCP	Transmission Control Protocol/Internet Protocol (TCP/IP)
WinSock UDP	UDP/IP

Installing the Controls

Methods to install the controls into the toolbox differ according to the application you are using.

In **Visual Basic**:

1. On the **Tools** menu,, click **Custom Controls**.
2. From Available Controls, select the control you want to use.
3. Click OK.

In **Visual FoxPro**:

1. On the Tools menu, click Options.
2. Choose the Controls tab, then click the OLE controls option. Be sure the Controls check box is selected.
3. Under Selected, select the controls you wish to install.
4. Click Set as Default if you'd like to make these controls available in every Visual FoxPro session.
5. Click OK..

In **Microsoft Access**:

While in form Design view:

1. On the Insert menu, click Custom Control…
2. Select the custom control you want to insert.
3. Click OK.

If the custom control does not appear in the custom control dialog, it may not be correctly registered on your machine. To register a custom control:

1. On the Tools menu, click Custom Controls.
2. Choose the Register button and locate the custom control .OCX or .DLL file to register.
3. Click OK.

In **Visual C++**:

Note: ActiveX controls are added to individual Visual C++ projects. To use an ActiveX control in multiple projects, you must add it to each project in which you want to use it.

1. From the Insert menu, choose Component, -- OR -- Click the Component Gallery button. The Component Gallery dialog box appears.
2. In Component Gallery, select the OLE Controls pane (page, tab).
3. Select the control and click the Insert button. The Confirm Classes dialog box appears. This dialog lists the names for the C++ class, header file, and implementation file for each control begin added.
4. In the Confirm Classes dialog box, click OK. The control is added to the control pallet in the dialog template editor, and header and implementation files are added to the current project for this control.

Common Terminology

The following is a short list of terms that occur in the documentation:

Client An application designed to connect to a remote machine and access data on the other machine. Part of a Client/Server relationship.

Data Streaming The transfer of data in a continuous stream of bytes, allowing transactions such as real-time sound transmission. Data streaming also allows you to efficiently pour data into a file, database, or other container.

MIME The Multipurpose Internet Mail Protocol communicates the type of data that is being sent in an electronic message. For example, a message can contain sound, video, images, as well as text.

Port Number Internet applications use a specific port number to communicate through. There are common port numbers assigned to types of applications. For example, HTTP servers (used for retrieving HTML documents) usually send and receive data through port 80.

Protocol A defined convention used to communicate between applications. For example, the HTTP protocol specifies how HTTP messages should be sent and received.

RFC (**Request For Comments)** document (as in "RFC 977"). A document authored by an individual or group involved in the development of the Internet. Such documents define standards or propose new standards for Internet protocols. These documents are available for downloading from various sites. See the Internet References topic for specific FTP addresses.

Server An application that provides services and data for a Client application that resides on a remote machine.

Control Architecture

The Internet ActiveX Controls are designed as a family of controls with a few common interfaces that allow you to develop applications quickly. Famliarizing yourself with these common interfaces can save you time later.

DocInput and DocOuput Objects

Features of the Internet ActiveX Controls are the **DocInput** and **DocOutput** objects. These objects, accessible from all except the WinSock controls, have properties and methods that allow you to stream data from one control to another. For example, using the **DocOutput** object, you can automatically stream data from a HTTP server to an **FTP** control when you wish to archive the data on an FTP server.

For more information See "Using the **DocInput** and **DocOutput** Objects."

The icError Object and Collection

Another common object is the **icError** object and **icErrors** collection. The **icError** object stores error messages that originate from a network server. As errors may be numerous, the **icError** object can be stored in a collection for later retrieval.

For more information See "**icError** Object" and "**icErrors** Collection."

DocHeader Object and Collection

Some protocols, such as the MIME and the SMTP, require you to create a collection of document headers. For example, when sending a mail message, you will see these headers:

```
From: Jonne@Maui.com
To: TomS@Haleakala.com
Subject: Run to the Sun
Message-Id: 9601138242.AA824256464@mail.Maui.com
```

Each line in the above example is contained in a single **DocHeader** object, and all four lines are part of a **DocHeaders** collection.

A **DocHeader** object has two properties: the **Name** property, and the **Value** property. The following

code shows how you can create a simple header by first creating an object variable of type DocHeaders, instantiating the object, and then adding a **DocHeader** object to the collection.

```
Dim dhHeads As DocHeaders ' Object variable.
Set dhHeads = New DocHeaders ' Instantiate object.
dhHeads.Add "From", "Jonne@Maui.com " ' Add Header.
' Creates the header "From: Jonne@Maui.com"
' The DocHeader object adds the colon for you.
```

For more information See "**DocHeader** Object" and "**DocHeaders** Collection."