CGI ActiveX Control Overview

The CGI ActiveX Control implements the Common Gateway Interface. CGI is a standard used for client applications to access information servers such as HTTP or Web servers. CGI programs are executed in real-time to output dynamic information. This differs from accessing normal HTML documents, which are static text files that don't change. Client applications, for example Web Browsers, can call CGI programs to retrieve dynamic data. For security reasons, CGI programs reside in a special directory,such as /cgi-bin, as specified by the server. This limits the number of programs and scripts that have access to the CGI.

When a client application is ready to call a CGI program, the application sets a series of environment variables which the CGI program can use. The CGI program may also get an additional input string based on the method set by the environment variable REQUEST_METHOD. If the request method is POST, then the input string is retrieved from standard input (file descriptor 0). If the request method is GET, then the input string is already placed in the environment variable QUERY_STRING. The input string is composed of zero or more input items separated by the '&' character (this is the standard CGI usage). Once the CGI program is finished processing the request, it sends the response to standard out (file descriptor 1), which will be transferred from the information server to the client application.

The main features of the CGI ActiveX Control are:

- The CGI variables supported by a Web Server can be accessed via the EnvironmentStrings collection.
- Other environment variables on the Server are also available in EnvironmentStrings.
- The input string (if URL-encoded) is parsed and stored in the QueryItems collection. If not URL-encoded, QueryItem is empty. The unparsed data is always stored in the QueryItemString property.
- Optional methods for decoding (Decode) and splitting strings (SplitString). Output is stored in DecodedString and SplitResults, respectively.
- The RequestMethod property indicates the kind of value set in the CGI variable REQUEST_METHOD.
- A response can be sent back to the client using either the SendDoc method or the SendReplyMessage() method.

CGI Properties, Methods, and Events

The following table lists the properties, methods, and events supported by the CGI ActiveX control. For an example illustrating the use of the control in a real life situation, see Handling Subscription Requests Sample.

Property	Method	Event
DecodedString	AboutBox	DocInput
DocInput	Decode	Error
EnvironmentStrings		
IsRawData	SendDoc	
QueryItem	SendReplyMessage	
QueryItemString	SplitString	
RequestMethod		
SplitResults		
EnvironmentStrings Collection		
Count	Item	
EnvironmentString Item		
Name		
Value		
QueryItems Collection		
Count	Item	
QueryItem Item		
Name		
Value		
SplitResults Collection		
Count	Item	
SplitResult Item		
Value		

Using the CGI Control

To use the CGI ActiveX Control you must choose the CGI toolbox icon. You must also use the correct syntax.

DecodedString CGI Property

Description

The results of the method Decode will be placed in this property.

Syntax

object.**DecodedString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

None.

Range

N/A.

DocInput CGI Property

Description

Object describing input information for the document being transferred.

Syntax

object.**DocInput**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

DocInput

Default Value

N/A.

Range

N/A.

Comments

The DocInput object provides a more powerful interface beyond the basic capabilities of the SendDoc method. For basic use of the control, knowledge or use of the DocInput object is not required.

Properties of the DocInput object may be set before calling the SendDoc method of the control, or they may be passed as arguments to this method. The DocInput object is also used for conveying information about the progress of the document transfer, for data linking and data streaming. For more information, see DocInput and DocOutput Objects.

EnvironmentStrings CGI Property

Description

A collection of EnvironmentString objects that contain environment settings, including CGI environment variables. This property may be indexed directly to retrieve an EnvironmentString object.

Syntax

object.**EnvironmentStrings**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

EnvironmentStrings.

Default Value

None.

Range

N/A.

Comments

A partial list of CGI environment variables are shown below**:**

Note: Not all environment variables are supported by all Web Servers (e.g. NetManage Personal Web Server, WebSite, NetSite, NCSA, CERN). Common to many Web Servers (Many but not all are supported by the Personal Web Server):

PATH_INFO	Path information that came along with the request.
PATH_TRANSLATED	Physical mappping that is derived from the virtual path given in PATH_INFO.
CONTENT_TYPE	For queries with attached information, such as those using the PUT method, the type of data attached.
CONTENT_LENGTH	Number of bytes of content being sent by the client.
GATEWAY_INTERFACE	Revision of the CGI specification this server complies. Format: CGI/revision (ex: CGI/1.1).
HTTP_ACCEPT	MIME types that the client will accept. Format: type/type,type/type,...
HTTP_USER_AGENT	Browser that the client is using.
QUERY_STRING	When a query URL or a form was sent using the GET method, the query information is stored here.
REMOTE_HOST	Hostname of machine making the request. Either the DNS name or alias.
REMOTE_ADDR	IP address of the REMOTE_HOST.
REQUEST_METHOD	The method that the request was made, either POST or GET.
SCRIPT_NAME	Virtual path to the script being executed.
SERVER_ADMIN	E-mail address of Web Administrator.
SERVER_NAME	Server's hostname, alias, or IP address.
SERVER_PORT	Port number server is accepting requests through (usually port 80).
SERVER_PROTOCOL	Name and revision of the information protocol this request came with. Format: protocol/revision (ex: HTTP/3.0).
SERVER_SOFTWARE	Name and version number of the information server software answering the request and running the gateway. Format: name/version (ex: Chameleon/6.0).

Other variables supported by one or more Web Servers (all supported by Personal Web Server):

DATE_GMT	Greenwich Mean Time date and time. Format: DAY, DD MM YYYY -- HH:MM:SS GMT ex: " WED, 02 10 1995 -- 14:35:15 GMT".
DATE_LOCAL	Local date and time with offset from GMT (for Pacific Standard Time, it is -0700) ex: "WED, 02 10 1995 -- 7:35:15 -0700".
DOCUMENT_ROOT	Physical path to the root of the Web Server.
GMT_OFFSET	The offset from GMT in seconds. ex: 25200 (for Pacific Standard Time).
HTTP_FROM	E-mail address of requester.
LOG_HTTP	Location of the HTTP log file for use in reporting statistics.
LOG_FTP	Location of the FTP log file for use in reporting statistics.
QUERY_STRING_UNESCAPED	QUERY_STRING with escaped characters translated to their ASCII values.
SERVER_ROOT	Logical path to the root of the Web Server.

Currently not supported by Personal Web Server but available on other servers:

AUTH_TYPE	Authentication method used to validate users for protected scripts.
REMOTE_USER	User name making the request Set only if user authentication has been used.
REMOTE_IDENT	User ID for a remote user in some authentication schemes.

IsRawData CGI Property

Description

This value is True if QueryItemString is not URL-encoded.

Syntax

object.**IsRawData**.

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

None.

Range

N/A

Comments

This property will be set to False if QueryItemString is URL-encoded (standard for form-based requests), and True if not. When IsRawData is True, QueryItems Collection will be empty. To get the raw data, you may use QueryItemString. URL_encoded input data should have the environment variable CONTENT_TYPE set to "application/x-www-form-urlencoded."

QueryItems CGI Property

Description

A collection of QueryItem objects sent from the client.

Syntax

object.**QueryItems**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

QueryItems.

Default Value

None.

Range

N/A.

Comments

If the InputString is URL-encoded (e.g., the CONTENT_TYPE is “application/x-www-form-urlencoded”) then this string is parsed into items in the QueryItems collection. Otherwise, there are no items in the collection and the raw input data (unparsed) can be accessed via the QueryItemString property.

QueryItemString CGI Property

Description

Unparsed input string as received from the client.

Syntax

object.**QueryItemString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

None.

Range

N/A.

Comment

This input string is sent from the client to the Web server and contains query information for the CGI application to process and respond to. If the CONTENT_TYPE indicates that the data is URL-encoded, this input string will also be parsed and placed in the QueryItems collection.

RequestMethod CGI Property

Description

Type of request.

Syntax

object.**RequestMethod**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

RequestMethodConstants.

Default Value

None.

Range

Name	Value	Description
icNameRequest	0	The environment variable REQUEST_METHOD is not defined.
IcGetRequest	1	The environment variable REQUEST_METHOD = GET.
IcPostRequest	2	The environment variable REQUEST_METHOD = POST.
IcOtherRequest	3	The environment variable REQUEST_METHOD is defined but is neither GET nor POST.

Comments

The actual REQUEST_METHOD can also be found by accessing the EnvironmentStrings collection item, indexed with the name of “REQUEST_METHOD.”

SplitResults CGI Property

Description

A collection of SplitResult Items. The results of the method SplitString will be placed in this property.

Syntax

object.**SplitResults**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

SplitResults collection.

Default Value

None.

Range

N/A.

Comments

Please refer to the SplitResults collection for details on accessing individual items in the SplitResults collection.

EnvironmentStrings Collection

A collection of EnvironmentString objects that contain environment settings, including CGI environment variables.

Count EnvironmentStrings Collection Property

Description

The number of attributes in the collection.

Syntax

object.**Count**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

None.

Range

N/A.

EnvironmentString Item

The EnvironmentString object is an item in a EnvironmentStrings collection. EnvironmentString items are used to specify the attribute names and values of the environment setting.

Name EnvironmentString Item Property

Description

The attribute name. This string is never empty. The Name associated with the item in the collection. (ex: for environment setting “REQUEST_METHOD=POST”, the Name would be “REQUEST_METHOD”).

Syntax

object.**Name**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

None.

Range

N/A.

Value EnvironmentString Item Property

Description

The Value associated with the environment variable. (ex: for environment setting "REQUEST_METHOD=POST" , the Value would be "POST").

Syntax

object.**Value**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

None.

Range

N/A.

QueryItems Collection

The QueryItems object is a collection containing QueryItem items.

Count QueryItems Collection Property

Description

The number of attributes in the collection. If the input data is a URL-encoded string, each element is input data sent by the client application (i.e. the Web browser). If input data is not URL-encoded, then QueryItems should be empty.

Syntax

object.**Count**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

None.

Range

N/A.

QueryItem Item

The QueryItem object is an item in a QueryItems collection.

Name QueryItem Item Property

Description

The element attribute name. (ex: for URL-encoded data, if a element is “FROMNAME=Bob”, then the Name would be “FROMNAME”).

Syntax

object.**Name**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

None.

Range

N/A.

Value QueryItem Item Property

Description

The element attribute value. (ex: for URL-encoded data, if a element is “FROMNAME=Bob”, then the Value would be “Bob”). The items in the list have had their escape characters removed and replaced with their ASCII equivalents.

Syntax

object.**Value**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

None.

Range

N/A.

SplitResults Collection

The SplitResults object is a collection containing SplitResult items.

Count SplitResults Collection Property

Description

The number of attributes in the collection.

Syntax

object.**Count**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

None.

Range

N/A.

SplitResult Item

The SplitResult object is an item in a SplitResults collection.

Value SplitResults Item Property

Description

The attribute value.

Syntax

object.**Value**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

None.

Range

N/A.

AboutBox CGI Method

Description

Shows information about this control.

Return Value

Void.

Syntax

object.**AboutBox**

Parameters

None.

Decode CGI Method

Description

Optional method used to remove escape characters from input string and replace them with its ASCII equivalent. Also replaces the '+' chars with space characters. The results of the Decode method will be placed in the property DecodedString. This method is not needed if the information provided in QueryItems and EnvironmentStrings is sufficient.

Return Value

Boolean.

Syntax

object.**Decode** (InputString **As String**)

Parameters

InputString

String to be decoded.

Data Type: String

Param: IN

Default Value: N/A.

SendDoc CGI Method

Description

Requests sending a document either in the format of a file or InputString.

Return Value

Void.

Syntax

object.**SendDoc** [*URL*], [*Headers*], [*InputData*], [*InputFile*], [*OutputFile*]

Parameters

URL

Optional. The URL identifying the remote document to be sent. Not used for the CGI version of SendDoc.

Data Type: String

Param: IN

Default Value: N/A.

Headers

Optional. Headers used for sending the document. This argument only applies to protocols where document headers can be sent (for example, SMTP and HTTP).

Data Type: DocHeaders

Param: IN

Default Value: DocInput.Headers

InputData

Optional. A data buffer containing the document to be sent. If input file is not empty, InputData will be ignored.

Data Type: VARIANT

Param: IN

Default Value: DocInput.GetData

InputFile

Optional. A local file containing the document to be sent.

Data Type: String

Param: IN

Default Value: DocInput.Filename

OutputFile

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).

Data Type: String

Param: IN

Default Value: DocOutput.Filename

Comments

The SendDoc method allows sending (posting or putting) a document. For each control, sending a document means something slightly different, for example in FTP it means putting a file onto the server, whereas in SMTP it means sending a message to the server.

The URL and (for some controls) Headers are used as inputs describing the document to be sent. For all controls, the InputData and InputFile arguments may contain the document to be sent (at most one of these may be specified). For controls like HTTP that return a reply document, the OutputFile argument may be used to indicate where the reply document should be written locally.

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, which are properties of this control. The properties of the DocInput and DocOutput objects can be set before calling SendDoc to avoid passing arguments. The DocInput and DocOutput events can also be used for transfering data using streaming rather than local files. See the DocInput and DocOutput properties, the DocInput and DocOutput events, and the separate DocInput and DocOutput object documentation for more information.

Note: URL and Headers parameters will be ignored for the CGI version of SendDoc.

SendReplyMessage CGI Method

Description

Sends a message back to the client. This message will be written to standard out, which is handled by the Web server calling the CGI program. You can either use this method or the SendDoc method to send out a response.

Return Value

Boolean.

Syntax

object.**SendReplyMessage** (ReplyMessage)

Parameters

ReplyMessage

The message to be sent.

Data Type: String

Param: IN

Default Value: None

SplitString CGI Method

Description

Optional method used to split up a string value StringToSplit which has zero or more a_separator characters inside of it. The result will be placed in the collection SplitResults. Each time this method is called, the data created from the previous SplitString request will be removed from the collection and the used memory will be freed. This method is not needed if the information provided in QueryItems and EnvironmentStrings is sufficient.

Return Value

Boolean.

Syntax

object.**SplitString** (StringToSplit **As VARIANT**, a_separator **As VARIANT**)

Parameters

StringToSplit

String value that needs to be split.

Data Type: VARIANT

Param: IN

Default Value: None

a_separator

The separator character

Data Type: VARIANT

Param: IN

Default Value: None

Item SplitResults Collection Method

Description

Returns an item from the collection based on the index.

Return Value

SplitResult.

Syntax

object.**SplitResults.Item** (Index **As VARIANT**)

Parameters

index

Index must be an integer. Integer indices identify an item by its 1-based index.

The Item method is the default method for a collection

Data Type: VARIANT

Param: IN

Default Value: None

Comment

The user should first check the Count property to see if there are any items available.

Item QueryItems Collection Method

Description

Returns an item from the collection based on the index.

Return Value

QueryItem.

Syntax

object.**QueryItems.Item** (Index **As VARIANT**)

Parameters

index

Index must be an integer. Integer indices identify an item by its 1-based index.

The Item method is the default method for a collection

Data Type: VARIANT

Param: IN

Default Value: None

Comment

First check the Count property of the collection to see if there are any Items available.

Item EnvironmentStrings Collection Method

Description

Returns an item from the collection based on the index.

Return Value

EnvironmentString. The index could also be a string value for the name. For example, to find the environment variable REQUEST_METHOD, use *object*.EnvironmentStrings.Item (“REQUEST_METHOD”)

Syntax

object.**EnvironmentStrings.Item** (Index **As VARIANT**)

Parameters

index

Index must be an integer. Integer indices identify an item by its 1-based index.

The Item method is the default method for a collection

Data Type: VARIANT

Param: IN

Default Value: None

Comment

The user should first check the Count property to see if there are any items available.

DocInput CGI Event

Description

A DocInput related event that indicates the input data has been transferred or the DocInput state has changed.

Syntax

*object*_**DocInput** (*DocInput* **As DocInput**)

Parameters

DocInput

Object describing document input data for the current transfer.

Data Type: DocInput

Param: IN

Default Value: N/A

Comments

For basic use of this control, the DocInput event can be used for notification of transfer progress, e.g., for updating a progress bar. The DocInput.BytesTotal, DocInput.BytesTransferred and DocInput.State properties can be examined in order to determine the current status of the transfer. For basic use, this event may be ignored if no progress information is needed.

For more powerful use of this control, the DocInput event can also be used for data streaming. See DocInput Object Overview.

Error CGI Event

Description

This event is activated when an error occurs in background processing (for example, failed to connect or failed to send or receive in the background).

Syntax

*object*_**Error** (*ErrCode* **As Integer**, *Description* **As String**, *Scode* **As Long**, *Source* **As String**, *HelpFile* **As String**, *HelpContext* **As Long**, *CancelDisplay* **As Boolean**)

Parameters

ErrCode

The short error code. For a list of possible CGI error codes see CGI Error Codes.

Description

String containing error information.

sCode

The long Scode.

Source

Error source.

HelpFile

Help file name.

HelpContext

Help file context.

CancelDisplay

Indicates whether to cancel the display. The default value for *CancelDisplay* is False meaning you do want to use the default message box. If you does not want to display the default error message box, set *CancelDisplay* to True.

Comments

If an error occurs, do NOT use a MessageBox to notify the user. Some information servers do not allow the MessageBox to appear. Also, this prevents the program from completing and returning a reply to the client, effectively causing the request to “hang.”

Handling Subscription Requests Sample

This sample shows how you can handle subscription requests from a Web Browser client. The user of the browser fills in information about themselves and submit that information by selecting a Submit button. This action causes an executable to be launched on the Web Server which can handle the request and can either accept, reject, or ask for more information by sending an HTML-formatted response back to the user.

The relevant portion of the HTML-formatted submission form is shown below (the Web Browser user will see prompts for Actual Name, User Name Requested, and Password Requested, as well as a Submit button):

```
<form method=POST action="/cgi-bin/wcgireq.exe">
<p>Actual Name:........<input name="ACTUALNAME" value="YourRealName"
   size="30">
<p>User Name Requested:.........<input name="USERNAME" value="YourUserName"
   size="30">
<p>Password Requested:.........<input name="PASSWD" value="YourPasswd"
   size="30">
<INPUT TYPE="Submit" VALUE="Submit">
```

This means that wcgireq.exe is executed by the Web Server once the Submit button is clicked. The entered information is sent to wcgireq.exe, an executable created in Visual Basic (or another ActiveX-enabled development environment) using the CGI NEWT Intranet ActiveX.

The CGI NEWT Intranet ActiveX gets the information and stores the user-entered data in the QueryItems collection. The CGI environment variables are stored in the EnvironmentStrings collection.

To access the data, the developer can either use a for loop or ask for the specific data based on the index (numeric or BSTR) to a variable:

```
‘example of using a for loop
Dim QueryItem As Object
For Each QueryItem In Wcgi1.QueryItems
   If QueryItems.Name = “USERNAME " Then
            <check to see if the username is already used>
      Elseif QueryItems.Name = “PASSWD” Then
            <check to see if the password is already used>
      End If
Next
```

Here is an example of using an index to access information in QueryItems:

```
     ‘the user created function ValidateUserName() expects
‘as input the UserName to be validated
     ValidateUserName (Wcgi1.QueryItems(“USERNAME”).Value)
```

Similarly, the EnvironmentStrings can be accessed in the same way. For example:

```
‘retrieves the Greenwich Mean Time
 Dim DateTime as String
DateTime = Wcgi1.EnvironmentStrings(“DATE_GMT”)
```

Since the Request Method of sending data is POST, the property RequestMethod is icPostRequest

The sample code below should be entered in a function called ProcessRequest in this example.

Inside of the ProcessRequest unction, the USERNAME and PASSWD are validated and a reply is formatted to be sent back.

```
Dim NL As String
Dim Title As String
Dim ReplyMsg As String
Dim gReplyMsg As String
```

```
NL = Chr(13) + Chr(10)  ‘ these are the /r/n characters
(validate the password and user name)

Title = "Response to Request"

If PasswordInvalid = TRUE Then
 gReplyMsg = “Sorry, the password you requested is not valid or in use.
              Please enter another one.” & NL
ElseIf NameInvalid = TRUE Then
 gReplyMsg = “Sorry, the Username you requested is not valid or in use.
              Please enter another one.” & NL
Else
gReplyMsg = (format a fancy reply welcoming the new user)
End If

ReplyMsg = "Content-type: text/html" & NL & NL
ReplyMsg = ReplyMsg & "<TITLE>" & Title & "</TITLE>" & NL
ReplyMsg = ReplyMsg & "<H2>" & Title & "</H2>" & NL & "<pre>" & NL & NL
ReplyMsg = ReplyMsg & NL & gReplyMsg
ReplyMsg = ReplyMsg & NL & "</pre>" & NL
```

To send the reply back, the developer can send it using the SendReplyMessage. method or SendDoc.

To use the SendReplyMessage() function, add the following code to ProcessRequest():

```
Wcgi1.SendReplyMessage ReplyMsg
```

To use the SendDoc() function, add the following code to ProcessRequest():

```
Wcgi1.SendDoc ,,ReplyMsg
```

Non-URL-Encoded Sample

But what if the information sent back is not-URL-encoded (i.e. the CGI variable CONTENT_TYPE is not "application/x-www-form-urlencoded")? Then Wcgi1.IsRawData=TRUE and there is no data in QueryItems. To clean up the data, the developer can use:

```
Wcgi1.Decode (Wcgi1.QueryItemString)
```

to replace escape characters in the QueryItemString with their ASCII equivalent, and also replace any '+' symbols with a space. The result of the decoding is placed in Wcgi1.DecodedString. Or the developer may also use:

```
Wcgi1.SplitString(Wcgi1.QueryItemString, "&")
' the "&" is an example of a separator character
```

to parse the information so that every piece of data in the string separated by the separator character is stored as an item in the SplitResults collection and can be processed individually.

FTP Client ActiveX Control Overview

The FTP (File Transfer Protocol) Client ActiveX Control allows files and data to be transferred between a remote and local machine. The next series of Help topics explain:

- The requirements for running the control
- A description of the properties, methods, and events
- An example illustrating the use of the control in a real life situation.

The FTP Client Control, invisible at run time to the user, provides easy access for Internet FTP services. It can be used by both Visual Basic, Delphi, and C++ programmers. To write applications that use FTP, you do not need to understand the details of FTP or how to call low-level WinSock APIs. By setting properties and calling methods on the control, you can easily send data to a remote machine and retrieve data from the network. Events are used to notify you of network activities.

The FTP Client Control also provides file and directory parsing of the List and NameList commands, making it possible for you to access the file size, attributes, name, or other fields without querying the server to determine the type of operating system (and hence file system) running on the remote server. For more details, see FTPDirItem Object.

Use of the FTP Control in Visual Basic gives you the ability to integrate file transfers into your program without learning a completely different computer language or transfer protocol. If you are a Visual Basic developer, you already have the knowledge to perform what would otherwise have been a complex task.

The following table summarizes the properties, methods, and events supported by the FTP Client Control. For an example illustrating the use of the control in a real life situation, see FTP Sample Session.

Property	Method	Event
AppendToFile	Abort	Abort
Blocking	AboutBox	Account
BlockResult	Account	Busy
Busy	Authenticate	Cancel
DocInput	Cancel	ChangeDir
DocOutput	ChangeDir	CreateDir
EnableTimer	Connect	DeleteDir
Errors	CreateDir	DelFile
Firewall	DeleteDir	DocInput
LocalPort	DeleteFile	DocOutput
Logging	Execute	Error
ListItemNotify	GetDoc	Execute
NotificationMode	GetFile	FirewallStateChanged
Operation	Help	Help
PassiveMode	List	ListItem
ProtocolState	Mode	Log
ProtocolStateString	NameList	Mode
RemoteFile	NOOP	NOOP
RemoteHost	ParentDir	ParentDir
RemotePort	PrintDir	PrintDir
ReplyCode	PutFile	ProtocolStateChanged
ReplyString	Quit	Quit
SleepTime	ReInitialize	ReInitialize
SocketHandle	SendDoc	Site
State	Site	StateChanged
StateString	System	System
Timeout	Type	TimeOut

URL	Type
UserId	

FTPDirItem Object

Attributes
Date
Detail
FileName
Size

Using the FTP Control

To use the FTP ActiveX Control you must choose the FTP toolbox icon and drag it into a Delphi/Visual Basic Form or a MSVC dialog resource.

FTP Property Pages

Property pages provide a standard interface for setting or accessing the properties exposed by a control. The following table lists the FTP properties that are listed on the property page.

Property Pages	**Properties Exposed**
Client Property Pages	NotificationMode FTP Property RemoteHost RemotePort
Authenticate Property Pages	Password UserId
General Property Pages	AppendToFile ListItemNotify

FTP Properties

Properties set the attributes for FTP Client behavior, although some of them may not have any affect until the client is connected to a server (for example, setting the AppendToFile property).

The following series of Help topics describe the properties supported by the FTP ActiveX Control. For an explanation of the description categories, see Object Descriptions.

AppendToFile FTP Property

Description

This property applies to PutFile and SendDoc to indicate whether the data should be appended to the file (True) or whether the file should be replaced (False).

Syntax

object.**AppendToFile** [= *Boolean*]

Permission

W (Read/Write).

Availability

D (Design)/Runtime.

Data Type

Boolean.

Default Value

False.

Range

True or False

Blocking FTP Property

Description

Indicates whether methods should block until complete or not.

Syntax

object.**Blocking** *[=Boolean]*

Permission

W (Read/Write)

Availability

D (Design) and R (Runtime)

Data Type

Boolean.

Default Value

False.

Range

True or False.

Comments

If Blocking is set to True, then a method will not be completed until the proper event(s) for a given request are fired, if applicable.

BlockResult FTP Property

Description

Returns the result value of the last blocking method called.

Syntax

object.**BlockResult**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

BlockingResultConstants.

Default Value

icBlockOK.

Range

Name	Value	Description
icBlockOK	0	Blocking method was successful.
IcTimedOut	1	Blocking method returned due to timeout.
IcErrorExit	2	Blocking method returned due to an error.
IcBlockCancel	3	Blocking method returned due to cancel.
IcUserQuit	4	Blocking method returned due application end.

Busy FTP Property

Description

Indicates that an operation/command is in progress.

Syntax

object.**Busy**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Boolean.

Default Value

N/A.

Range

True or False

DocInput FTP Property

Description

Object describing input information for the document being transferred.

Syntax

object.**DocInput**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

DocInput.

Default Value

N/A.

Range

N/A.

Comments

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

Properties of the DocInput object may be set before calling the SendDoc method or they may be passed as arguments to this method. The DocInput object is also used for conveying information about the progress of the document transfer and for data linking and streaming.

For more information, see Common Control Objects.

DocOutput FTP Property

Description

Object describing output information for the document being transferred.

Syntax

object.**DocOutput**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

DocOutput.

Default Value

N/A.

Range

N/A.

Comments

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

Properties of the DocOutput object may be set before calling the GetDoc method or they may be passed as arguments to this method. The DocOutput object is also used for conveying information about the progress of the document transfer, and for data linking and streaming.

For more information, see DocOutput Object Overview.

EnableTimer FTP Property

Description

Enable timer for the specified event. The event is specified by entering:

```
EnableTimer(short event)
```

Syntax

object.**EnableTimer** (*event*) [= *Boolean*]

Permission

W (Write Only).

Note: This is the only control property that is Write only.

Availability

R (Runtime)

Default Value

False. (The timer for this event will not be enabled.)

Range

True or False

Comments

Event is an integer value that determines the type of Timeout event that will be enabled. Constants defined for enum types for events are:

Value	Meaning
prcConnectTimeout = 1	Timeout for connect. If connection is not established within the timeout period, the Timeout event will be activated.
prcReceiveTimeout = 2	Timeout for receiving data. If no data arrives within the timeout period, the Timeout event will be activated.
prcUserTimeout= 65	Timeout for user defined event. Use prcUserTimeout + [Integer] range for custom timeout events.

Errors FTP Property

Description

A collection of errors that can be accessed for details about the last error that occurred. This collection should be used within an Error event if information passed through the Error event is not sufficient.

Syntax

object.**Errors**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

icErrors.

Default Value

N/A.

Range

N/A.

Firewall FTP Property

Description

Object describing proxy access to firewall FTP servers.

Syntax

object.**Firewall**.

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Firewall

Default Value

N/A.

Range

N/A.

Comments

When the Firewall object is used within the FTP client control, the FTP control is able to provide transparent access to hosts external to firewall-protected domains. You can set mode and host information used to connect to the external host through the Firewall object's properties.

LocalPort FTP Property

Description

Designates the local port to use.

Syntax

object.**LocalPort** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

0.

Range

0 - 65535

Logging FTP Property

Description

Indicates whether log events should be fired when log data is available.

Syntax

object.**Logging** [= *Boolean*]

Permission

W (Read/Write).

Availability

D (Design) and R(Runtime).

Data Type

Boolean.

Default Value

False.

Range

True or False

ListItemNotify FTP Property

Description

Causes the container to receive events for every directory element received during a List or NameList command.

Syntax

object.**ListItemNotify** [= *Boolean*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Boolean.

Default Value

False. (FTPDirItem objects are not parsed in the ListItem event.)

Range

True or False

Comments

If this property is TRUE, the directory listing is parsed and events activated for every directory element. If this property is FALSE, the list data is sent in blocks to the data target during ProcessData notifications.

For more information on directory parsing, see ListItem FTP Event. For more information on data streaming, see DocStream.

NotificationMode FTP Property

Description

Determines when notification is issued for incoming data. Notification can also be suspended.

Syntax

object.**NotificationMode** [= *Integer*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

NotificationModeConstants.

Default Value

icCompleteMode.

Range

0-maximum unsigned long. At present, the values are:

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

Operation FTP Property

Description

Allows you to determine which method caused data to be received. This property is normally used when processing the DocOutput event.

Syntax

object.**Operation**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

FTPOperationConstants.

Default Value

ftpList.

Range

- ftpFile= 0
- ftpList = 1
- ftpNameList = 2

Comments

The GetFile, GetDoc, List, and NameList methods all transfer data via the DocOutput event. Using this property allows you to determine which method activated the DocOutput event, making it possible to distinguish between the various types of data.

PassiveMode FTP Property

Description

Determines whether data correction will attempt to use server passive mode.

Syntax

object.**PassiveMode** [= *Boolean*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

Boolean.

Default Value

FALSE.

Range

TRUE or FALSE.

Password FTP Property

Description

Password of current user on the FTP Server.

Syntax

object.**Password** [= *String*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

String.

Default Value

Empty.

Range

N/A.

ProtocolState FTP Property

Description

This property specifies the current state of the protocol.

Syntax

object.**ProtocolState**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

FTPProtocolStateConstants.

Default Value

ftpBase.

Range

0-2. Constants defined for enum types of ProtocolState property are:

Value	**Meaning**
ftpBase = 0	Default. The state before connection server is established.
ftpAuthorization = 1	Authorization is performed.
ftpTransaction = 2	Authorization successful. The client has successfully identified itself to the FTP server.

ProtocolStateString FTP Property

Description

String representation of ProtocolState.

Syntax

object.**ProtocolStateString**

Permission

R (Read-only).

Availability

R (Runtime) and D (Design).

Data Type

String.

Default Value

“BASE”.

Range

N/A.

RemoteFile FTP Property

Description

The remote file name used during GetFile and PutFile operations.

Syntax

object.**RemoteFile** [= *String*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

RemoteHost FTP Property

Description

The remote machine to connect to if the RemoteHost parameter in the Connect method is missing. You can either provide a host name or an IP address string in dotted format. For example, “127.0.0.1”.

Syntax

object.**RemoteHost** [= *String*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

String.

Default Value

“127.0.0.1”

Range

N/A.

RemotePort FTP Property

Description

The remote port number to which to connect.

Syntax

object.**RemotePort** [= *Long*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

Long.

Default Value

21.

Range

1-65535.

ReplyCode FTP Property

Description

The value of the reply code is a protocol specific number that determines the result of the last request, as returned in the ReplyString property.

Syntax

object.**ReplyCode**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

0

Range

See RFC 959 for valid reply codes.

ReplyString FTP Property

Description

Lists the last reply string sent by the FTP Server to the client as a result of a request. This string contains both a number code and a status string that the server creates for the last command.

Syntax

object.**ReplyString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

SleepTime FTP Property

Description

Specifies the sleep time between checking messages, if Blocking is True.

Syntax

object.**SleepTime** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

10 ms.

Range

>=zero.

Comments

Only applies when in Blocking mode.

SocketHandle FTP Property

Description

Socket handle for the primary connection (Request/Reply connection).

Syntax

object.**SocketHandle**

Permission

R (Read only)

Availability

R (Runtime).

Data Type

Long.

Default Value

N/A.

Range

>=0

Comments

Some protocols require more than one connection. SocketHandle is the handle for the request/reply connection. If the value is less than zero, the SocketHandle is valid.

State FTP Property

Description

This property specifies the connection state of the control.

Syntax

object.**State**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

StateConstants.

Default Value

prcDisconnected.

Range

1-6. Constants defined for enum types of State property are:

Value	**Meaning**
prcConnecting = 1	Connecting. Connect has been requested, waiting for connect acknowledge.
prcResolvingHost = 2	Resolving Host. Occurs when RemoteHost is in name format rather than dot-delimited IP format.
prcHostResolved = 3	Resolved the host. Occurs only if ResolvingHost state has been entered previously.
prcConnected = 4	Connection established.
prcDisconnecting = 5	Connection closed. Disconnect has been initiated.
prcDisconnected = 6	Initial state when protocol object is instantiated, before Connect has been initiated, after a Connect attempt failed or after Disconnect performed.

StateString FTP Property

Description

A string representation of State.

Syntax

object.**StateString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

"Disconnected".

Range

N/A.

Timeout FTP Property

Description

Timeout value for the specified event. The event is specified by entering:

```
Timeout(short event)
```

Syntax

object.**Timeout** (*event*) [= *Long*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

Long.

Default Value

0.

Range

0-maximum unsigned long. Constants defined for enum types for events are:

Value	Meaning
prcConnectTimeout = 1	Timeout for connect. If connection is not established within the timeout period, the Timeout event will be activated.
prcReceiveTimeout = 2	Timeout for receiving data. If no data arrives within the timeout period, the Timeout event will be activated.
prcUserTimeout= 65	Timeout for user defined event. Use prcUserTimeout + [Integer] range for custom timeout events.

URL FTP Property

Description

URL (Universal Resource Locator) string identifying the current document being transferred. The URL format when using the FTP Control is:

```
FTP://username:password@host:port/documentnameandpath;type=type
```

where `type` is specified as ‘A’, ‘I’, or ‘D’. “A” is for ASCII, “I” is for Binary, and “D” is for Directory listing.

Syntax

object.**URL** [= S*tring*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

String.

Default Value

Empty string.

Range

Valid URL.

Comments

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 FTP control, the URL property identifies a remote file transferred via FTP. 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 `ftp` string may be omitted when using the FTP control.

UserId FTP Property

Description

User identification name for the client on the server.

Syntax

object.**UserId** [= *String*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

String.

Default Value

Empty.

Range

N/A.

FTP Methods

Methods are called to perform a particular operation on an object. After the method is successfully processed, you will receive an event with a name similar to the method called. You can then check the ReplyCode for the server response or check error codes if an error message is generated.

For a list of FTP methods, see FTP Client ActiveX Control Overview. For an explanation of the description categories, see Object Descriptions.

Abort FTP Method

Description

Requests a FTP Server to abort the last data transfer request. Similar to the FTP RFC-959 ABORT command.

Return Value

Void.

Syntax

object.**Abort**

Parameters

None.

Comments

This event usually terminates any data connection while leaving the control connection intact.

AboutBox FTP Method

Description

Shows information about this control.

Return Value

Void.

Syntax

object.**AboutBox**

Parameters

None.

Account FTP Method

Description

Sends account information to remote host. Similar to the FTP RFC-959 ACCT command.

Return Value

Void.

Syntax

object.**Account** *Account*

Parameters

Account

String containing new account information.

Data Type: String

Param: Out

Default Value: Empty

Comments

Use the ReplyString property to determine the result of this call. The Account event is fired if successful.

Authenticate FTP Method

Description

Authenticates the user based on the parameters passed. If no parameters are passed, the UserId and Password properties are used. If neither the UserId or Password is entered, the control uses the URL.

Return Value

Void.

Syntax

object.**Authenticate** [*UserID*], [*Password*]

Parameters

UserId

Optional. User identification string to use for authentication.

Data Type: String

Param: IN

Default Value: N/A

Password

Optional. Password to use for authentication.

Data Type: String

Param: IN

Default Value: N/A

Comments

If the UserId and/or Password are set before invoking this method, the optional parameters do not need to be specified. Optional arguments to this method override the values from corresponding UserId and Password properties. If you omit one or both of the arguments, the value from a corresponding property will be used to provide the authentication. The state will then be an FTP transaction. The ProtocolStateChanged event is fired if successful. The event will be an FTP transaction.

Cancel FTP Method

Description

Cancels a pending request and disconnects the current session.

Return Value

Void.

Syntax

object.**Cancel**

Parameters

None.

ChangeDir FTP Method

Description

Requests FTP Server to change the remote host current directory to the specified directory. Similar to the FTP RFC-959 CWD command.

Return Value

Void.

Syntax

object.**ChangeDir** *directory*

Parameters

Directory

String containing new directory name.

Data Type: String

Param: IN

Default Value: Empty

Comments

Use the ReplyString property to determine the result of this call. If successful, the ChangeDir event will fire.

Connect FTP Method

Description

Initiates a Connect request. The control calls the StateChanged event and Protocol StateChanged event if a connection is established.

Return Value

Void.

Syntax

object.**Connect** [*RemoteHost*], [*RemotePort*]

Parameters

RemoteHost

Optional. Remote host to which to connect. If this parameter is missing, the control connects to the host defined in the RemoteHost property.

Data Type: String

Param: IN

Default Value: N/A

RemotePort

Optional. Remote port to which to connect. If this parameter is missing, the control connects to the port defined in the RemotePort property.

Data Type: Long

Param: IN

Default Value: N/A

Comments

Optional arguments to this method override the values from corresponding RemoteHost and RemotePort properties. If no argument is given, the values from the properties will be used to establish the connection.

CreateDir FTP Method

Description

Creates the specified directory on the remote host. Similar to the FTP RFC-959 MKD command.

Return Value

Void.

Syntax

object.**CreateDir** *Directory*

Parameters

Directory

String containing the directory name.

Data Type: String

Default Value: Empty

Comments

Use the ReplyString property to determine the result of this call. If successful, the CreateDir event will fire.

DeleteDir FTP Method

Description

Deletes the specified directory file from the remote host. Similar to the FTP RFC-959 RMD command.

Return Value

Void.

Syntax

object.**DeleteDir** *Directory*

Parameters

Directory

String containing directory to delete.

Data Type: String

Param: IN

Default Value: Empty

Comments

Use the ReplyString property to determine the result of this call. If successful, the DeleteDir event will fire.

DeleteFile FTP Method

Description

Deletes the specified file from the remote host. Similar to the FTP RFC-959 DELE command.

Return Value

Void.

Syntax

object.**DeleteFile** [*FileName*]

Parameters

FileName

Optional. String containing file to delete. This argument overrides the values from the corresponding RemoteFile property. The value of the property will not change. If the argument is omitted, the value from the corresponding property will be used to provide a filename for the delete operation.

Data Type: String

Param: IN

Default Value: Empty

Comments

Use the ReplyString property to determine the result of this call. If successful, the DeleteFile event will fire.

Execute FTP Method

Description

Issues the RFC 959 Quote command to the server.

Return Value

Void.

Syntax

object.**Execute** *cmd*

Parameters

Cmd

String containing the command to be invoked on the remote FTP server. This forces the FTP server to process a specific command(s) without actually issuing the command from the client.

Data Type: String

Param: IN

Default Value: Empty

Comments

Use the ReplyString property to check the return value during the Execute event notification. If successful, the Execute event will fire.

GetDoc FTP Method

Description

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

Return Value

Void.

Syntax

object.**GetDoc** [*URL*], [*OutputFile*]

Parameters

URL

Optional. The URL identifying the remote document to be retrieved.

Data Type: String

Param: IN

Default Value: DocInput.URL

OutputFile

Optional. A local file to which the retrieved document will be written.

Data Type: String

Param: IN

Default Value: DocOutput.FileName

Comments

The GetDoc method in FTP means retrieving a file or directory listing 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) may be omitted and will default to the correct type for this control. For example, when using the FTP control, the "ftp:" string may 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.

{button ,KL(`DocInput and DocOutput Objects',0,`',`')} **See Also**

GetFile FTP Method

Description

Gets the specified file from the remote host and places it in the current directory.

Return Value

Void.

Syntax

object.**GetFile** [*RemoteFile*], [*LocalFile*]

Parameters

RemoteFile

Optional. String containing the remote file to retrieve.

Data Type: String

Param: IN

Default Value: Empty

LocalFile

Optional. String containing the local filename to use when saving the remote file.

Data Type: String

Param: IN

Default Value: Empty

Comments

The arguments override the values from the corresponding FTP.RemoteFile and DocOutput.FileName properties. The value of the properties will not change. If the arguments are omitted, the value from the corresponding property will be used to provide a filenames for the get file operation.

Both local and remote names should be specified, even if they are the same.

Use the ReplyString to determine the result of this call. The data from this method is sent to the DocStream interface via the DocOutput event. During processing of the DocOutput event, the Operation property is set to FTPFile.

Help FTP Method

Description

Gets FTP help from the remote host. Similar to the FTP RFC-959 HELP command.

Return Value

Void.

Syntax

object.**Help** *Help*

Parameters

Help

String containing Help Commands supported

Data Type: String

Param: IN

Default Value: Empty

Comments

Use the ReplyString property to determine the result of this call. If successful, the Help event will fire.

List FTP Method

Description

Requests a detailed directory listing of the specified directory from the remote host. Similar to the FTP RFC-959 LST command.

Return Value

Void.

Syntax

object.**List** [*Directory]*

Parameters

Directory

Optional. String containing path of remote host from which to list directories.

Data Type: String

Default Value: Empty

Comments

The data from this method is sent to the DocStream interface via the DocOutput event. During processing of the DocOutput event, the Operation property is set to ftpList. If the ListItemNotify property is set to True, the ListItem event is also generated for every item in the directory listing.

Mode FTP Method

Description

Sets data transfer mode of remote host. Similar to the FTP RFC-959 MODE command.

Return Value

Void.

Syntax

object.**Mode** *Mode*

Parameters

Mode

Enumerated type containing new transfer mode information The FTPModeConstants may have one of the following values.

- ftpStream = 0
- ftpBlock = 1
- ftpCompressed = 2

Data Type: FTPModeConstants

Param: IN

Default Value: Empty

Comments

If successful, the Mode event will fire.

NameList FTP Method

Description

Requests a directory listing of the specified directory from the remote host. Similar to the FTP RFC 959 NLST command.

Return Value

Void.

Syntax

object.**NameList** *[Directory]*

Parameters

Directory

Optional. String containing remote host path from which to list directories.

Data Type: String

Param: IN

Default Value: Empty

Comments

The data from this method is sent to the DocStream interface via the DocOutput event. During processing of the DocOutput event, the Operation property is set to ftpNameList. If the ListItemNotify property is set to True, the ListItem event is also generated for every item in the directory listing.

NOOP FTP Method

Description

Issues the NOOP command to the server.

Return Value

Void.

Syntax

object.**NOOP**

Parameters

None.

Comments

Use the ReplyString property to determine the result of this call. If successful, the NOOP event will fire.

ParentDir FTP Method

Description

Requests the FTP Server change to the parent of the current directory, if one exists.

Return Value

Void.

Syntax

object.**ParentDir**

Parameters

None.

Comments

Use the ReplyString property to determine the result of this call. If successful, the ParentDir event will fire.

PrintDir FTP Method

Description

Requests the FTP Server query the current directory of the remote host. Similar to the FTP RFC-959 PWD command.

Return Value

Void.

Syntax

object.**PrintDir**

Parameters

None.

Comments

Use the ReplyString property to determine the result of this call. You will need to parse the ReplyString to determine the directory name. You can also obtain this information from the RemoteDir property. If successful, the PrintDir event will fire.

PutFile FTP Method

Description

Puts specified file on the Server's current directory.

Return Value

Void.

Syntax

object.**PutFile** [*LocalFile*], [*RemoteFile*]

Parameters

LocalFile

Optional. String containing the name of the local file to be transferred to the remote machine.

Data Type: String

Param: IN

Default Value: Empty

RemoteFile

Optional. String containing name of file to be placed on the remote machine.

Data Type: String

Param: IN

Default Value: Empty

Comments

The arguments override the values from the corresponding FTP.RemoteFile and DocInput.FileName properties. The value of the properties will not change. If the arguments are omitted, the value from the corresponding property will be used to provide a filenames for the put file operation.

Both local and remote names should be specified, even if they are the same.

Use the ReplyString property to determine the result of this call during DocInput event processing. The DocStream DocInput event is activated when file data is streamed out of the FTP Control.

Quit FTP Method

Description

Quits a session with remote host and terminates any data connection.

Return Value

Void.

Syntax

object.**Quit**

Parameters

None.

ReInitialize FTP Method

Description

Issues the ReInit command to the server.

Return Value

Void.

Syntax

object.**ReInitialize**

Parameters

None.

Comments

Use the ReplyString property to determine the result of this call. If successful, the ReInitialize event will fire.

SendDoc FTP Method

Description

A DocInput related method that requests sending a document identified by a URL, InputFile, or InputData.

Return Value

Void.

Syntax

object.**SendDoc** [*URL*], [*Headers*], [*InputData*], [*InputFile*], [*OutputFile*]

Parameters

URL

Optional. The URL identifying the remote document to be sent. If specified, the URL property will be set to this value.

Data Type: String

Param: IN

Default Value: DocInput.URL

Headers

Optional. Headers used for sending the document. This argument only applies to protocols where document headers can be sent (for example, SMTP and HTTP).

Data Type: DocHeaders

Param: IN

Default Value: DocInput.Headers

InputData

Optional. A data buffer containing the document to be sent.

Data Type: VARIANT

Param: IN

Default Value: DocInput.GetData

InputFile

Optional. A local file containing the document to be sent.

Data Type: String

Param: IN

Default Value: DocInput.FileName

OutputFile

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).

Data Type: String

Param: IN

Default Value: DocOutput.FileName

Comments

The SendDoc method in FTP means putting a file on 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 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) may be omitted and will default to the correct type for this control. For example, when using the FTP control, the "ftp:" string may 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.

{button ,KL(`DocInput and DocOutput Objects',0,`',`')} **See Also**

Site FTP Method

Description

Issues a Site command to the remote server. This command is used during logon to determine the file system supported on the server.

Return Value

Void.

Syntax

object.**Site** [*Sitecmd]*

Parameters

Sitecmd

Optional. String containing Site command line options to be processed by the server (for example, “DIRSTYLE”)

Data Type: String

Param: IN

Default Value: Empty

Comments

Use the ReplyString property to determine the result of this call. If successful, the Site event will fire.

Status FTP Method

Description

Requests status from the remote host. Similar to the FTP RFC-959 STAT command.

Return Value

Void.

Syntax

object.**Status** [*Status]*

Parameters

Status

Optional. String containing status string.

Data Type: String

Param: OUT

Default Value: Empty

Comments

Use the ReplyString property to determine the result of this call during the Status event notification.

System FTP Method

Description

Issues a system command to the remote server. It is similar to the FTP RFC-959 SYST command.

Return Value

Void.

Syntax

object.**System**

Parameters

None.

Comments

Use the ReplyString property to determine the result of this call. If successful, the System event will fire.

Type FTP Method

Description

Issues a Type command to the remote server. This command is entered prior to a data transfer to set the transfer type.

Return Value

Void.

Syntax

object.**Type** *ftpType*

Parameters

ftpType

Enumerated type containing type of data. The server attempts to use this value for data representation if it is supported. Possible values are:

- ftpAscii = 0
- ftpEBCDIC = 1
- ftpImage = 2
- ftpBinary = 3

Data Type: FTPTypeConstants

Param: IN

Default Value: ftpAscii

Comments

If successful, the Type event will fire.

FTP Events

Events are used for FTP client notification. They indicate that an action has been requested and processed. Any errors which occur during command processing result in the Error event being called with appropriate error codes. Error codes, state changes, and protocol return values are usually checked during event processing. (See icErrors Collection Overview.)

Almost all FTP Control methods have an associated event that is activated after the server processes the command (either successfully or unsuccessfully). During the event notification of any given command, you need to check the ReplyString property to determine the outcome of the command. For example, you may call the Site method, and receive a Site event. During notification of the Site event, you would check the ReplyString property to see the result of the Site command.

The FTP methods GetFile, GetDoc, List, and NameList all retrieve data and share the same DocOutput event notification. As a developer you need to know which of the operations caused the event and redirect the data to the appropriate place (i.e., send List data to a list box control, and send file data to an edit control for display.) To do this, check the Operation property to determine the activating event.

The following Help topics describe the events supported by the FTP Client Control. Each description includes the syntax, related parameters, their data type, default value, and whether the parameter is used for input or output (IN or OUT). For a complete list of FTP events, see FTP Client ActiveX Control Overview.

Abort FTP Event

Description

This event is activated after the Abort method is called. It aborts any active data connection process, if supported by the server.

Syntax

*object*_**Abort**

Parameters

None.

Account FTP Event

Description

This event is activated after the Account method is called. It requests the remote host set the account to the one specified.

Syntax

*object*_**Account**

Parameters

None.

Comments

Use the ReplyString property to determine the server reply after event processing.

Busy FTP Event

Description

This event is activated when a command is in progress or when a command has completed.

Syntax

*object*_**Busy** (*Busy* **As Boolean**)

Parameters

Busy

Indicates whether or not a command is in progress.

Data Type: Boolean. If the argument is True, a command is in progress.

Cancel FTP Event

Description

This event is activated after a cancellation request has been completed and satisfied. After this event the object's state changes to Base.

Syntax

*object*_**Cancel**

Parameters

None.

ChangeDir FTP Event

Description

This event is activated after the ChangeDir method is called. It changes the current working directory.

Syntax

*object*_**ChangeDir**

Parameters

None.

Comments

Use the ReplyString property to determine the server reply after event processing.

CreateDir FTP Event

Description

This event is activated after the CreateDir method is called. It creates a new directory.

Syntax

*object*_**CreateDir**

Parameters

None.

DeleteDir FTP Event

Description

This event is activated after the DeleteDir method is called. It deletes the specified directory on the remote host.

Syntax

*object*_**DeleteDir**

Parameters

None.

Comments

Use the ReplyString property to determine the server reply after event processing.

DelFile FTP Event

Description

This event is activated after the DeleteFile method is called. It deletes the specified file located in the path specified on the remote host.

Syntax

*object*_**DelFile**

Parameters

None.

DocInput FTP Event

Description

A DocInput related event that indicates the input data has been transferred, or the DocInput state has changed.

Syntax

*object*_**DocInput** (*DocInput* **As DocInput**)

Parameters

DocInput

Object describing document input data for the current transfer.

Data Type: DocInput

Param: IN

Default Value: N/A

Comments

The DocInput event can be used in its basic form for notification of transfer progress, (for example, for updating a progress bar). The DocInput.BytesTotal, DocInput.BytesTransferred and DocInput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

To gain more power from this control, you can also use the DocInput event for data streaming. See DocInput Object Events.

DocOutput FTP Event

Description

A DocOutput related event indicating that output data has been transferred or the DocOutput state has changed.

Syntax

*object*_**DocOutput** (*DocOutput* **As DocOutput**)

Parameters

DocOutput

Object describing document output data for the current transfer.

Data Type: DocOutput

Param: IN

Default Value: N/A

Comments

The DocOutput event can be used in its basic form to notify the user of transfer progress, (for example, for updating a progress bar). The DocOutput.BytesTotal, DocOutput.BytesTransferred and DocOutput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

To gain more power from this control, you can also use the DocOutput event for data streaming. For more information, see DocInput Object Events.

Error FTP Event

Description

This event is activated when an error occurs in background processing (for example, failed to connect or failed to send or receive in the background).

Syntax

*object*_**Error** (*ErrCode* **As Integer**, *Description* **As String**, *Scode* **As Long**, *Source* **As String**, *HelpFile* **As String**, *HelpContext* **As Long**, *CancelDisplay* **As Boolean**)

Parameters

ErrCode

The short error code. For a list of possible FTP error codes see FTP Error Codes.

Description

String containing error information.

sCode

The long Scode.

Source

Error source.

HelpFile

Help file name.

HelpContext

Help file context.

CancelDisplay

Indicates whether to cancel the display. The default is TRUE (no display of the default error message box). If you do want to use the default message box, set CancelDisplay to FALSE.

Execute FTP Event

Description

This event is activated after the Execute method is called. It issues a command to the server for processing.

Syntax

*object*_**Execute**

Parameters

None.

Comments

Use the ReplyString property to determine the results of the Execute command during event notification.

FirewallStateChanged FTP Event

Description

This event is activated whenever the state of the firewall connection state changes.

Syntax

*object.***FirewallStateChanged** (FirewallState As Integer)

Parameters

Refer to the Firewall object FirewallState property and FirewallStateString for possible values of the FirewallState parameter.

Help FTP Event

Description

This event is activated after the HELP method is called. It requests the remote host send help information with the specified HELP parameters.

Syntax

*object*_**Help**

Parameters

None.

Comments

Use the ReplyString property to determine the server reply after event processing.

ListItem FTP Event

Description

This event is activated for every element in a directory listing when the ListItemNotify property is set to TRUE. This lets you parse the directory elements after issuing a List or NameList command.

See FTPDirItem Object.

Syntax

*object*_**ListItem** (*Item* **As ftpDirItem**)

Parameters

Item

Object describing a directory listing element. The object contains the filename, size, date, and attributes of the current listing item.

Data Type: ftpDirItem

Param: OUT

Default Value: N/A

Log FTP Event

Description

This event is fired when logging data is available.

Syntax

*object*_**Log**

Parameters

None.

Mode FTP Event

Description

This event is activated after the Mode method is called. It sets the remote host data transfer mode to the mode specified.

Syntax

*object*_**Mode**

Parameters

None.

Comments

Use the ReplyString property to determine the response from the server.

NOOP FTP Event

Description

This event is activated after a NOOP command is issued or the NOOP method is called. It requests an OK reply from the server.

Syntax

*object*_**NOOP**

Parameters

None.

Comments

Use the ReplyString property to determine the response from the server.

ParentDir FTP Event

Description

This event is activated after the ParentDir method is called. It changes the current directory on the remote host to the parent directory, if one exists.

Syntax

*object*_**ParentDir**

Parameters

None.

Comments

Use the ReplyString property to determine the server reply after event processing.

PrintDir FTP Event

Description

This event is activated after the PrintDir method is called. It requests the remote host include the current working path as a string in the reply.

Syntax

*object*_**PrintDir**

Parameters

None.

ProtocolStateChanged FTP Event

Description

This event is activated whenever the protocol state changes.

Syntax

*object*_**ProtocolStateChanged** (*ProtocolState* **As Integer**)

Parameters

Refer to the ProtocolState property and ProtocolStateString for possible values of the ProtocolState parameter.

ReInitialize FTP Event

Description

This event is activated after the ReInitialize method is called. It causes the server to log off the current user while maintaining an open connection.

Syntax

*object*_**ReInitialize**.

Parameters

None.

Comments

Use the ReplyString property to determine the result of the Reinitialize method during event notification.

Site FTP Event

Description

This event is activated after the Site method is called. It requests directory and file formatting information from the remote host.

Syntax

*object*_**Site**

Parameters

None.

Comments

Use the ReplyString property to determine the server reply.

StateChanged FTP Event

Description

This event is activated whenever the state of the transport state changes.

Syntax

*object*_**StateChanged** (*State* **As Integer**)

Parameters

Refer to the State property and StateString for possible values of the state parameter.

System FTP Event

Description

This event is activated after the System method is called. It requests the type of operating system on the server.

Syntax

*object*_**System**

Parameters

None.

Comments

Use the ReplyString property to determine the server reply.

TimeOut FTP Event

Description

This event is activated when the timer for the specified event has expired.

Syntax

*object*_**TimeOut** (ByVal *Event* **As Integer,** *Continue* **As Boolean**)

Parameters

Event

Defines the event to which the time interval applies.

Data Type: Short

Continue

Determines if the timer is active or not. Set *Continue* to TRUE to keep the timer active.

Data Type: Boolean

Default Value: False

Comments

Event is an integer value that determines the type of Timeout event that will be enabled. Constants defined for enum types for events are:

Value	**Meaning**
prcConnectTimeout = 1	Timeout for connect. If connection is not established within the timeout period, the Timeout event will be activated.
prcReceiveTimeout = 2	Timeout for receiving data. If no data arrives within the timeout period, the Timeout event will be activated.
prcUserTimeout= 65	Timeout for user defined event. Use prcUserTimeout + [Integer] range for custom timeout events.

See Also

Timeout FTP Property.

Type FTP Event

Description

This event is activated after the Type method is called. Specifies how the remote host should handle the transferred data.

Syntax

*object*_**Type**

Parameters

None.

Comments

Use the ReplyString property to determine the server reply after event processing.

Firewall FTP Object

The Firewall object provides an interface to get and set information used to access hosts external to a firewall-protected domain. Through this interface, you can set a few parameters and then access external hosts as you would if the firewall did not exist.

Properties

Host

Mode

Password

Port

State

StateString

UserId

Host Firewall FTP Property

Description

The address or name of the remote machine that is the firewall server. You can either provide a host name or an IP address string in dotted format. For example, “127.0.0.1”.

Syntax

object.Firewall.Host [= String]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

String.

Default Value

“127.0.0.1”

Range

N/A.

Mode Firewall FTP Property

Description

Determines what method is to be used by the FTP control to connect to external hosts through a firewall server.

Syntax

object.Firewall.Mode [= Integer]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

FTPFirewall Mode constants.

Default Value

ftpFirewallOff.

Range

At present, the values are:

Name	Constant	Meaning
ftpFirewallOff	0	Firewall service is not used.
ftpFirewallSite	1	After the connection and login is made to the firewall server, the external connection to the FTP.RemoteHost is made using the SITE command.
FtpFirewallOpen	2	After the connection and login is made to the firewall server, the external connection to the FTP.RemoteHost is made using the OPEN command.
FtpFirewallUserLogin	3	After the connection and login is made to the firewall server, the external connection to the FTP.RemoteHost is made using the USER <user@hostname> command.
FtpFirewallUserNoLogin	4	After the connection is made to the firewall server, the external connection to the FTP.RemoteHost is made using the USER <user@hostname> command.

Password Firewall FTP Property

Description

Password of Firewall.UserId on the FTP Firewall Server. This value is only used if the mode of the Firewall requires password verification.

Syntax

object.**Firewall.Password** [= String]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

Port Firewall FTP Property

Description

The port number of the Firewall server host to connect to.

Syntax

object.**Firewall.Port** [= Long]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

Long.

Default Value

21.

Range

1-65535.

Comment

Although the data type is long, Winsock conventions limit the maximum correct port number to be the maximum signed short value.

State Firewall FTP Property

Description

This property specifies the connection state of the firewall session.

Syntax

object.Firewall.State

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

FTPFirewall Mode constants

Default Value

ftpFirewallBase.

Range

1300-1306. Constants defined for enum types of FTP Firewall State property are:

Name	Value	Meaning
ftpFirewallBase	1300	There is no current connection in progress
FtpFirewallConnected	1301	The control is connected to the firewall server.
ftpFirewallConnected_Msg	1302	The control is connected to the firewall server and the server hello message has been received.
ftpFirewallUser_OK	1303	The control is connected to the firewall server and the UserId is valid.
ftpFirewallAuthorized	1304	The control is connected to the firewall server and the user has been authenticated to the firewall server.
FtpFirewallRemotedConnected	1305	The control is connected to the firewall server and the connection to the FTP.RemoteHost has been made.
ftpFirewallDisconnecting	1306	The control is in the process of disconnecting from the external host.

StateString Firewall FTP Property

Description

A string representation of the Firewall State.

Syntax

object.**Firewall.StateString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

"Firewall base state: Disconnected".

Range

N/A.

UserId Firewall FTP Property

Description

User identification name for the client on the Firewall server. This value is only used if the mode of the Firewall requires user authentication.

Syntax

object.Firewall.UserId [= String]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

FTPDirItem Object

The FTPDirItem object is used for directory and file parsing. Generally when a List command is processed, the server returns a byte stream whose format is dependent on the operating system running on the remote server.

Directory parsing is turned on by setting the ListItemNotify property to TRUE. Whenever a List command is processed, the FTP Control generates ListItem events for every directory or file element in the listing. The parameter list of the ListItem event includes an FTPDirItem object which contains the filename, size, date, and attributes for the current listing element.

If the ListItemNotify property is set to FALSE; the directory listing is sent to the Document Output object which in turn sends event notifications as streams of data are received. This is useful when you want to put directory listing data into a static control or an edit field in an ‘as-is’ state.

FTPDirItem Object Example

The following example demonstrates how to use the FTPDirItem object with directory parsing.

1. Set the ListItemNotify property to TRUE.
   ```
   ftpct1.ListItemNotify = TRUE
   ```
2. Invoke a List or NameList command
   ```
   ftpct1.List
   ```
3. As you receive the ListItem events, the item parameter contains the current directory element :

```
Private Sub Ftpct1_OnListItem(ByVal FTPDirItem As Object)
  Form2.text1.Text = Form2.text1.Text & Chr$(13) & Chr$(10) &
  FTPDirItem.FileName & Chr$(9) & FTPDirItem.Size & Chr$(9) & FTPDirItem.Date &
  Chr$(9) & FTPDirItem.Attributes & Chr$(9) & Chr$(9) & FTPDirItem.Detail
End Sub
```

FTPDirItem Object Properties

FTPDirItem object properties let you access the filename, date, file size, and file attributes of a directory listing. For more information on each of these properties, click on an item below.

Property

Attributes
Date
Detail
FileName
Size

Attributes FTPDirItem Property

Description

File or directory attributes of the current directory listing item.

Syntax

object.**Attributes**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Integer.

Default Value

0.

Range

This value is an integer value that may be any of the following

Attribute	Value
FtpUnknown	0x0000
FtpDir	0x0001
FtpFile	0x0002
FtpLink	0x0010
FtpDisk	0x0004

Date FTPDirItem Property

Description

Last modified date of the current directory listing item.

Syntax

object.**Date**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

Detail FTPDirItem Property

Description

Contains the raw unparsed data that the FTP server sends back for this directory item.

Syntax

object.**Detail**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

FileName FTPDirItem Property

Description

File or directory name of the current directory listing item.

Syntax

object.**FileName**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

Size FTPDirItem Property

Description

File or directory name of the current directory listing item.

Syntax

object.**Size**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Long.

Default Value

0.

Range

>=0.

FTP Localization

The resources for the control's about box, property page, and strings are in resource DLL nmorenu.dll. The resource DLL is localized for each language.

FTP Sample Session

This simple application demonstrates the use of the FTP Client Control in an every day work situation.

Every Monday, Mr. B must submit a report of all his employees weekly goals and accomplishments. Before FTP Control, he would manually request these documents. With Microsoft Visual Basics, he is able to create a form that takes blocks of formatted data and puts it into an easy to read summary. For each employee, Mr. B embeds an FTP Control into the form causing the control to download the employee's documents from a predefined directory on the employee's local system. Mr. B's Visual Basic code would look like this:

```
Ftpctl1.RemoteHost="Dilbert"
Ftpctl1.UserID="MrBig"
Ftpctl1.Password="0123"
FtpCtl1.RemoteFile="c:\\docs\\Goals.doc";
               OR
FtpCtl1.URL="FTP://MrBig:0123@Dilbert/c|/docs/Goals.doc";

FtpCtl1. DocOutput.FileName="Dilbert.doc"
FtpCtl1.GetFile()
     :

     :
Ftpctl2.RemoteHost="Dogbert"
Ftpctl2.UserID="MrBig"
Ftpctl2.Password="0123"

Rem This time the remote filename is specified in the parameter Rem list of the
method

FtpCtl2.DocOutput.FileName ="Dogbert.doc"

FtpCtl2.GetFile Goals.doc
```

Instead of embedding an FTP Control into each individual file, Mr. B could have used a single FTP Control and iterated through a list of engineers and individually requested the files from each user.

HTML ActiveX Control Overview

The HTML control lets you implement an HTML viewer, with or without automatic network retrieval of HTML documents. It provides parsing and layout of HTML data, as well as a scrollable view of the selected HTML page. It can also be used as a non-visual HTML parser to analyze or process HTML documents.

See Also

HTML Properties, Methods, and Events
HTML ActiveX Features
Retrieving HTML Data
DocStreams
Non-Visual HTML Parser
Using the HTML Control
HTML Limitations

HTML ActiveX Features

The HTML control supports the following features:

- Scrollable view of selected page
- Inline graphics: GIF, JPEG, BMP, XBM
- HTML version 2.X plus most NetScape 2.0 and Explorer 2.0 extensions
- Built-in document retrieval for HTTP and File URLs
- Built-in HTTP form execution
- Properties controlling the style sheet (such as fonts and colors)
- DocStream interfaces for flexible data transfer
- Events for overriding default processing
- Printing

See Also

HTML Properties, Methods, and Events
Retrieving HTML Data
DocStreams
Non-Visual HTML Parser
Using the HTML Control
HTML Limitations

Retrieving HTML Data

HTML source text or graphics data can be retrieved in the following ways:

- Explicitly. You can call the RequestDoc and RequestSubmit methods to explicitly specify a new main document by URL or request submission of a form. These methods cause the DoRequestDoc and DoRequestSubmit events to be activated.
- By selecting an active link. When you click on an active link, a request retrieval is made of a new main document, identified by the URL of the link. The default for this request is to retrieve the document using HTTP or a local file. If the request is successful, the DoRequestDoc event will be activated.
- By selecting embedded documents that are to be displayed inline. If successful, the DoRequestEmbedded event will be activated. The default for this request is to retrieve the document using HTTP or from a local file.
- By requesting via form submission. When you click on a form submission button, the DoRequestSubmit event is activated after a successful form submission. The response is used as the next main document.

See Also

HTML Properties, Methods, and Events
DocStreams
Non-Visual HTML Parser
Using the HTML Control
HTML Limitations

DocStreams

DocStream (DocInput and DocOutput) objects are the mechanism used for data retrieval and submission. You can use the DoRequestDoc, DoRequestEmbedded and DoRequestSubmit events to view or modify a DocStream for document retrieval or submission. By default, DocStream objects are created internally by the HTML control for documents with HTTP and File URLs. To provide for other URL types, or specify a DocStream object for any URL type, you can set DocStream properties during event handling.

See Also

HTML Properties, Methods, and Events
HTML ActiveX Features
Retrieving HTML Data
Non-Visual HTML Parser
Using the HTML Control
HTML Limitations

Non-Visual HTML Parser

The HTML Control can also be used as a non-visual HTML parser. If the control is set to be invisible at run-time, no view window is created. When HTML input data is processed and the ElemNotification property is set to True, the DoNewElement event is activated as each element is parsed. You can query the attributes and values of the parsed element when DoNewElement is activated. If this event is canceled, parsing will continue but the HTML Control will not store the element.

See Also

HTML Properties, Methods, and Events
HTML ActiveX Features
Retrieving HTML Data
DocStreams
Using the HTML Control
HTML Limitations

Using the HTML Control

To use the HTML ActiveX Control you must choose the HTML toolbox icon.

When using the HTML control with built-in network document retrieval, or when linking it to other network controls for document retrieval, there should be no added overhead for transfer of data between objects, i.e., there should be no copying of data and notifications should perform as well as ordinary C++ function calls.

No event handling should be necessary to implement a simple Web viewer with browsing and form submission capabilities when using the basic features of the HTML control. For more powerful use of the control, you can override all built-in document retrieval, browser and form submission behavior.

When using the control as a non-visual parser, there should be no overhead for visual (dormant) aspects of the control.

The HTML control requires dual (direct call) OLE interfaces. It also uses and is dependent on the DocInput, DocOutput and DocHeader[s] objects.

See Also

HTML Properties, Methods, and Events
HTML ActiveX Features
Retrieving HTML Data
DocStreams
Non-Visual HTML Parser
HTML Limitations

HTML Limitations

The following features are not supported by the HTML control:

- Text selection and clipboard copy
- Automatic external viewer launching
- Proxy server determination and usage
- Built-in FTP retrieval and inline FTP listings
- Basic Authorization and SSL/PCT
- Multipart document submission (file upload)

See Also

HTML Properties, Methods, and Events
HTML ActiveX Features
Retrieving HTML Data
DocStreams
Non-Visual HTML Parser
Using the HTML Control

HTML Properties, Methods and Events

The following table lists the properties, methods and events supported by the HTML control. For an example illustrating the use of the control in a real life situation, see HTML Sample Sessions.

Property	Method	Event
BackColor	AboutBox	BeginRetrieval
BackImage	AutoPrint	Click
BaseURL	BeginPrinting	DblClick
Blocking	Cancel	DocInput
BlockResult	EndPrinting	DocOutput
DeferRetrieval	GetPlainText	DoNewElement
DocBackColor	IsprintingDone	DoRequestDoc
DocForeColor	PrintPage	DoRequestEmbedded
DocInput	RequestAllEmbedded	DoRequestSubmit
DocLinkColor	RequestDoc	EndRetrieval
DocOutput	SelectAll	Error
DocVisitedColor		GotFocus
Errors		KeyDown
ElemNotification		KeyPress
FixedFont		KeyUp
Font		LayoutComplete
ForeColor		LostFocus
Forms		MouseDown
HasSelection		MouseMove
Heading1Font		MouseUp
Heading2Font		ParseComplete
Heading3Font		TimeOut
Heading4Font		UpdateRetrieval
Heading5Font		
Heading6Font		
hWnd		
IsPrintingDone		
LayoutDone		
LinkColor		
ParseDone		
Redraw		
RequestURL		
RetainSource		
RetrieveBytesDone		
RetrieveBytesTotal		
SleepTime		
SourceText		
Timeout		
TotalHeight		
TotalWidth		
UnderlineLinks		
URL		
UseDocColors		

ViewSource	
VisitedColor	
HTMLAttrs Collection	Item
Count	
HTMLAttr Item	
Name	
Value	
HTMLForms Collection	Item
Count	
HTMLForm Item	RequestSubmit
Method	
URL	
URLEncodedBody	

BackColor HTML Property

Description

Defines the default background color.

Syntax

object.**BackColor** [= *color*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long (OLE_COLOR).

Default Value

ActiveX container's BackColor

Range

RGB color values: 0 to 16777215 (0xFFFFFF).

Comments

May be overridden by the DocBackColor property, if such a document color is present and the UseDocColors property is True.

An RGB color is a mixture of the percentages of red, green, and blue, each component having a value between 0 and 255 (e.g., 0x00 and 0xFF). An RGB value of (0,0,0) (e.g., 0x000000 hex, or 0 decimal) produces black, while an RGB value of (255,255,255) (e.g., 0xFFFFFF hex or 16777215 decimal) produces white. For more details, please refer to the Visual Basic Help documentation on BackColor and ForeColor properties.

BackImage HTML Property

Description

URL of an image to be used as the background image of the document.

Syntax

object.**BackImage** [= *String*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

String.

Default Value

Empty string.

Range

Valid URL.

Comments

May be overridden by the background image of the document (<BODY BACKGROUND=...>) if this attribute is present and the UseDocColors property is True. The background image is tiled to fill the view area of the control window.

BaseURL HTML Property

Description

URL of the <BASE> element of the current document, used for relative URL resolution. If no <BASE> element exists in the document, this property is the same as the URL property.

Syntax

object.**BaseURL**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty string.

Range

Valid URL.

Comments

If no <BASE> element exists in the document, this property is the same as the URL property.

Blocking HTML Property

Description

Indicates whether methods should block until complete or not.

Syntax

object.**Blocking** *[=Boolean]*

Permission

W (Read/Write)

Availability

D (Design) and R (Runtime)

Data Type

Boolean.

Default Value

False.

Range

True or False.

Comments

If Blocking is set to True, then a method will not be completed until the proper event(s) for a given request are fired, if applicable.

BlockResult HTML Property

Description

Returns the result value of the last blocking method called.

Syntax

object.**BlockResult**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

BlockingResultConstants.

Default Value

icBlockOK.

Range

Name	Value	Description
icBlockOK	0	Blocking method was successful.
IcTimedOut	1	Blocking method returned due to timeout.
IcErrorExit	2	Blocking method returned due to an error.
IcBlockCancel	3	Blocking method returned due to cancel.
IcUserQuit	4	Blocking method returned due application end.

Comments

If Blocking is set to True, then a method will not be completed until the proper event(s) for a given request are fired, if applicable.

DeferRetrieval HTML Property

Description

Indicates whether retrieval of embedded objects should be deferred until explicitly requested.

Syntax

object.**DeferRetrieval** [= *Boolean*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Boolean.

Default Value

False.

Range

True or False.

Comments

The user can set this property to turn inline retrieval of embedded documents off or on. If you are implementing caching, you will normally leave this property set to False so that cached documents are always displayed inline.

DocBackColor HTML Property

Description

Document background color.

Syntax

object.**DocBackColor** [= *color*]

Permission

R (Read Only).

Availability

R (Runtime).

Data Type

Long (OLE_COLOR).

Default Value

object.**BackColor**.

Range

RGB color values: 0 to 16777215 (0xFFFFFF).

Comments

This property corresponds to the BGCOLOR attribute of the BODY tag. If this attribute is not present, HTML defaults to the value of the BackColor property.

An RGB color is a mixture of the percentages of red, green, and blue, each component having a value between 0 and 255 (e.g., 0x00 and 0xFF). An RGB value of (0,0,0) (e.g., 0x000000 hex, or 0 decimal) produces black, while an RGB value of (255,255,255) (e.g., 0xFFFFFF hex or 16777215 decimal) produces white. For more details, please refer to the Visual Basic Help documentation on BackColor and ForeColor properties.

DocForeColor HTML Property

Description

Document foreground (text) color.

Syntax

*object***.DocForeColor** [=c*olor*]

Permission

R (Read Only).

Availability

R (Runtime).

Data Type

Long (OLE_COLOR)..

Default Value

*object.***ForeColor**.

Range

RGB color values: 0 to 16777215 (0xFFFFFF).

Comments

This property corresponds to the TEXT attribute of the BODY tag. If this attribute is not present, HTML defaults to the value of the ForeColor property.

An RGB color is a mixture of the percentages of red, green, and blue, each component having a value between 0 and 255 (e.g., 0x00 and 0xFF). An RGB value of (0,0,0) (e.g., 0x000000 hex, or 0 decimal) produces black, while an RGB value of (255,255,255) (e.g., 0xFFFFFF hex or 16777215 decimal) produces white. For more details, please refer to the Visual Basic Help documentation on BackColor and ForeColor properties.

DocInput HTML Property

Description

Object describing input information for the main document being transferred.

Syntax

object.**DocInput**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

DocInput.

Default Value

N/A.

Range

N/A.

Comments

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

DocLinkColor HTML Property

Description

Document link color.

Syntax

object.**DocLinkColor** [= *color*]

Permission

R (Read-only)

Availability

R (Runtime).

Data Type

Long (OLE_COLOR)..

Default Value

*object.*LinkColor.

Range

RGB color values: 0 to 16777215 (0xFFFFFF).

Comments

This property corresponds to the LINK attribute of the BODY tag. If this attribute is not present, HTML defaults to the value of the LinkColor property.

An RGB color is a mixture of the percentages of red, green, and blue, each component having a value between 0 and 255 (e.g., 0x00 and 0xFF). An RGB value of (0,0,0) (e.g., 0x000000 hex, or 0 decimal) produces black, while an RGB value of (255,255,255) (e.g., 0xFFFFFF hex or 16777215 decimal) produces white. For more details, please refer to the Visual Basic Help documentation on BackColor and ForeColor properties.

DocOutput HTML Property

Description

Object describing output information when submitting form data.

Syntax

object.**DocOutput**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

DocOutput.

Default Value

N/A.

Range

N/A.

Comments

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

See Also

See the Form.RequestSubmit method, the DoRequestSubmit event, the DocOutput event and the DocOutput Object section of the Common Control Objects chapter for more information

DocVisitedColor HTML Property

Description

Document visited link color.

Syntax

object.**DocVisitedColor** [= *color*]

Permission

R (Read-only)

Availability

R (Runtime).

Data Type

Long (OLE_COLOR)..

Default Value

*object.*VisitedColor.

Range

RGB color values: 0 to 16777215 (0xFFFFFF).

Comments

This property corresponds to the VLINK attribute of the BODY tag. If this attribute is not present, HTML defaults to the value of the VisitedColor property.

An RGB color is a mixture of the percentages of red, green, and blue, each component having a value between 0 and 255 (e.g., 0x00 and 0xFF). An RGB value of (0,0,0) (e.g., 0x000000 hex, or 0 decimal) produces black, while an RGB value of (255,255,255) (e.g., 0xFFFFFF hex or 16777215 decimal) produces white. For more details, please refer to the Visual Basic Help documentation on BackColor and ForeColor properties.

ElemNotification HTML Property

Description

Indicates whether the DoNewElement event should be activated during HTML parsing.

Syntax

object.**ElemNotification** [= *Boolean*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Boolean.

Default Value

False.

Range

True or False.

Comments

You can set this property to True when using the HTML control as a (visual or nonvisual) parser.

See Also

DoNewElement event

Errors HTML Property

Description

A collection of errors that can be accessed for details about the last error that occurred. This collection should be used within an Error event if information passed through the Error event is not sufficient.

Syntax

object.**Errors**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

icErrors.

Default Value

N/A.

Range

N/A.

FixedFont HTML Property

Description

The Font object for fixed-width text.

Syntax

object.**FixedFont**

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Font.

Default Value

Courier New, size 10.

Range

Valid font.

Comments

There are several properties of the Font object, such as Name, Size, and Bold, which modify the type of font being used. Please refer to the Visual Basic Help documentation on the Font object for more information.

Font HTML Property

Description

The Font object for fixed-width text.

Syntax

object.**Font**

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Font.

Default Value

Times New Roman, size 12.

Range

Valid font.

Comments

There are several properties of the Font object, such as Name, Size, and Bold, which modify the type of font being used. Please refer to the Visual Basic Help documentation on the Font object for more information.

ForeColor HTML Property

Description

Default foreground (text) color.

Syntax

object.**ForeColor** [= *color*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long (OLE_COLOR)..

Default Value

ActiveX container's ForeColor.

Range

RGB color values: 0 to 16777215 (0xFFFFFF).

Comments

This property may be overridden by the DocForeColor property if such a document color is present and the UseDocColors property is True.

An RGB color is a mixture of the percentages of red, green, and blue, each component having a value between 0 and 255 (e.g., 0x00 and 0xFF). An RGB value of (0,0,0) (e.g., 0x000000 hex, or 0 decimal) produces black, while an RGB value of (255,255,255) (e.g., 0xFFFFFF hex or 16777215 decimal) produces white. For more details, please refer to the Visual Basic Help documentation on BackColor and ForeColor properties.

Forms HTML Property

Description

A collection of the forms contained in the HTML page.

Syntax

object.**Forms**

Permission

R (Read-only)

Availability

R (Runtime).

Data Type

HTMLForms.

Default Value

None.

Range

None.

Comments

This property may be indexed directly to call the default Item method. See section HTML Forms Collection later in this chapter for more information.

HasSelection HTML Property

Description

Returns whether a text selection exists.

Syntax

object.**HasSelection** [=Boolea*n*]

Permission

R (Read Only).

Availability

R (Runtime).

Data Type

Boolean.

Default Value

N/A.

Range

True/False.

Heading1Font HTML Property

Description

The Font object for heading level 1 text (<H1> elements).

Syntax

object.**Heading1Font**

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Font.

Default Value

Times New Roman, size 24, Bold.

Range

Valid font.

Comments

There are several properties of the Font object, such as Name, Size, and Bold, which modify the type of font being used. Please refer to the Visual Basic Help documentation on the Font object for more information.

Heading2Font HTML Property

Description

The Font object for heading level 2 text (<H2> elements).

Syntax

object.**Heading2Font**

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Font.

Default Value

Times New Roman, size 18, Bold.

Range

Valid font.

Comments

There are several properties of the Font object, such as Name, Size, and Bold, which modify the type of font being used. Please refer to the Visual Basic Help documentation on the Font object for more information.

Heading3Font HTML Property

Description

The Font object for heading level 3 text (<H3> elements).

Syntax

object.**Heading3Font**

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Font.

Default Value

Times New Roman, size 14, Bold.

Range

Valid font.

Comments

There are several properties of the Font object, such as Name, Size, and Bold, which modify the type of font being used. Please refer to the Visual Basic Help documentation on the Font object for more information.

Heading4Font HTML Property

Description

The Font object for heading level 4 text (<H4> elements).

Syntax

object.**Heading4Font**

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Font.

Default Value

Times New Roman, size 12, Bold.

Range

Valid font.

Comments

There are several properties of the Font object, such as Name, Size, and Bold, which modify the type of font being used. Please refer to the Visual Basic Help documentation on the Font object for more information.

Heading5Font HTML Property

Description

The Font object for heading level 5 text (<H5> elements).

Syntax

object.**Heading5Font**

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Font.

Default Value

Times New Roman, size 10, Bold.

Range

Valid font.

Comments

There are several properties of the Font object, such as Name, Size, and Bold, which modify the type of font being used. Please refer to the Visual Basic Help documentation on the Font object for more information.

Heading6Font HTML Property

Description

The Font object for heading level 6 text (<H6> elements).

Syntax

object.**Heading6Font**

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Font.

Default Value

Times New Roman, size 8, Bold.

Range

Valid font.

Comments

There are several properties of the Font object, such as Name, Size, and Bold, which modify the type of font being used. Please refer to the Visual Basic Help documentation on the Font object for more information.

hWnd HTML Property

Description

The handle of the HTML view window. This is an advanced property. It could be used to post messages like WM_COPY, etc..

Syntax

object.**hWnd**

Permission

R (Read Only).

Availability

R (Runtime).

Data Type

HANDLE.

Default Value

N/A.

Range

N/A.

IsPrintingDone HTML Property

Description

Used to determine the end of the document.

Return Value

Boolean.

Syntax

object.**IsPrintingDone** *page*

Permission

R (Read-only)

Availability

R (Runtime).

Parameters

Page

This value should be 1 greater than the last page printed.

Data Type: Long

Param: IN

Default Value: None

Comments

IsPrintingDone checks whether the document is finished printing or whether additional pages needs to be printed.

LayoutDone HTML Property

Description

Indicates whether the layout phase is complete.

Syntax

object.**LayoutDone**

Permission

R (Read-only)

Availability

R (Runtime).

Data Type

Boolean.

Default Value

False.

Range

True or False.

Comments

This property is set to False when document retrieval starts, and set to True when layout (placement of items on the page) of the main document is complete.

LinkColor HTML Property

Description

Default link color.

Syntax

object.**LinkColor** [= *color*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long (OLE_COLOR)..

Default Value

Blue (0,0,255)

Range

RGB color values: 0 to 16777215 (0xFFFFFF).

Comments

This property may be overridden by the DocLinkColor property if such a document color is present and the UseDocColors property is True.

An RGB color is a mixture of the percentages of red, green, and blue, each component having a value between 0 and 255 (e.g., 0x00 and 0xFF). An RGB value of (0,0,0) (e.g., 0x000000 hex, or 0 decimal) produces black, while an RGB value of (255,255,255) (e.g., 0xFFFFFF hex or 16777215 decimal) produces white. For more details, please refer to the Visual Basic Help documentation on BackColor and ForeColor properties.

ParseDone HTML Property

Description

Indicates whether the parsing phase is complete.

Syntax

object.**ParseDone**

Permission

R (Read-only)

Availability

R (Runtime).

Data Type

Boolean.

Default Value

False.

Range

True or False.

Comments

This property is set to False when document retrieval starts, and set to True when parsing of the main document is complete.

Redraw HTML Property

Description

Indicates whether drawing should occur as data changes or the window is scrolled.

Syntax

object.**Redraw** [= *Boolean*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Boolean.

Default Value

True.

Range

True or False.

Comments

To make changes and avoid flickering (redrawing when each change is made), set the Redraw property to False, make the changes, and then set it back to True. When Redraw is set to True, the window will be redrawn.

RequestURL HTML Property

Description

URL string identifying the new document requested.

Syntax

object.**RequestURL**

Permission

R (Read Only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty String.

Range

Valid URL.

Comments

You can specify this property by calling RequestDoc. The property is set by the control during default processing for the DoRequestDoc event.

RetainSource HTML Property

Description

Indicates whether source text should be retained and available via the SourceText property.

Syntax

object.**RetainSource** [= *Boolean*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Boolean.

Default Value

True.

Range

True or False.

Comments

This property may be set to False to save memory when you do not need the source text of the main document.

RetrieveBytesDone HTML Property

Description

Completed byte size of the objects being retrieved. This property is zero if no retrieval is in progress.

Syntax

object.**RetrieveBytesDone**

Permission

R (Read-only)

Availability

R (Runtime).

Data Type

Long.

Default Value

Zero.

Range

>=zero.

RetrieveBytesTotal HTML Property

Description

Total byte size of the objects to be retrieved, including embedded objects and the document itself.

Syntax

object.**RetrieveBytesTotal**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

Zero.

Range

>=zero.

Comments

If DeferRetrieval is set to True, RetrieveBytesTotal does not include embedded objects. This value can change during retrieval as object sizes are determined. This property is zero if no retrieval is in progress.

SleepTime HTML Property

Description

Specifies the sleep time between checking messages while Blocking is True.

Syntax

object.**SleepTime** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

10 ms.

Range

>=zero.

Comments

Only applies when in Blocking mode.

SourceText HTML Property

Description

Contains the source text of the main document.

Syntax

object.**SourceText**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

None.

Range

None.

Comments

This property will be empty if the RetainSource property is False or if no main document has been retrieved.

TimeOut HTML Property

Description

Time-out interval (in seconds) for initiating the request for documents. The Timeout event is activated if no data is received within timeout.

Syntax

object.**Timeout** [= *Long*]

Permission

W (Read/Write).

Availability

D (design) and R (Runtime).

Data Type

Long.

Default Value

30 seconds.

Range

N/A.

Comments

Although the Timeout value applies to all document retrieval, the Timeout event is activated only for the main document, not for embedded documents.

Event is an integer value that determines the type of Timeout event that will be enabled. Constants defined for enum types for events are:

Value	Meaning
prcConnectTimeout = 1	Timeout for connect. If connection is not established within the timeout period, the Timeout event will be activated.
prcReceiveTimeout = 2	Timeout for receiving data. If no data arrives within the timeout period, the Timeout event will be activated.
prcUserTimeout= 65	Timeout for user defined event. Use prcUserTimeout + [Integer] range for custom timeout events.

TotalHeight HTML Property

Description

Total height of the document in pixels.

Syntax

object.**TotalHeight**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

Zero.

Range

>=zero.

Comments

This property reflects the total height of the document, including the area that may not be visible because the view is smaller than the document. This property is updated as parsing and layout of the HTML document occurs. Its value is final when the EndRetrieval event is activated.

TotalWidth HTML Property

Description

Total width of the document in pixels.

Syntax

object.**TotalWidth**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

Zero.

Range

>=zero.

Comments

This property reflects the total width of the document, including the area that may not be visible because the view is smaller than the document. This property is updated as parsing and layout of the HTML document occurs. Its value is final when the EndRetrieval event is activated.

UnderlineLinks HTML Property

Description

Indicates whether links should be underlined.

Syntax

object.**UnderlineLinks** [= *Boolean*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Boolean.

Default Value

True.

Range

True or False.

URL HTML Property

Description

URL string identifying the current main document.

Syntax

object.**URL**

Permission

R (Read Only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty String.

Range

Valid URL.

Comments

This property is set by the control from the RequestURL property when document retrieval has successfully started and the BeginRetrieval event is activated.

UseDocColors HTML Property

Description

Indicates whether document colors should be used when present.

Syntax

object.**UseDocColors** [= *Boolean*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Boolean.

Default Value

True.

Range

True or False.

Comments

If this property value is True, the document colors (if present) override the default colors. For example, if the <BODY LINK=...> attribute is present and UseDocColors is True, then the color specified for the LINK attribute will be used to display active links; otherwise, the LinkColor property value will be used.

ViewSource HTML Property

Description

Indicates whether the control should display HTML source as plain text.

Syntax

object.**ViewSource** [= *Boolean*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Boolean.

Default Value

False.

Range

True or False.

Comments

This property is set to True to view the source text of the main document. If this property is True and RetainSource is False, document retrieval will be initiated to obtain the source text for viewing.

VisibleColor HTML Property

Description

Determines whether a view window is visible at runtime.

Syntax

object.**Visible** [= *Boolean*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Boolean.

Default Value

True.

Range

True or False.

Comments

This property may be overridden by the DocVisitedColor property if such a document color is present and the UseDocColors property is True.

VisitedColor HTML Property

Description

Default visited link color.

Syntax

object.**VisitedColor** [= *color*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long (OLE_COLOR).

Default Value

Purple (255,0,255)

Range

Valid color.

Comments

This property may be overridden by the DocVisitedColor property if such a document color is present and the UseDocColors property is True.

An RGB color is a mixture of the percentages of red, green, and blue, each component having a value between 0 and 255 (e.g., 0x00 and 0xFF). An RGB value of (0,0,0) (e.g., 0x000000 hex, or 0 decimal) produces black, while an RGB value of (255,255,255) (e.g., 0xFFFFFF hex or 16777215 decimal) produces white. For more details, please refer to the Visual Basic Help documentation on BackColor and ForeColor properties.

AboutBox HTML Method

Description

Shows information about this control.

Return Value

Void.

Syntax

object.**AboutBox**

Parameters

None.

AutoPrint HTML Method

Description

AutoPrint prints the entire HTML document including inserting page breaks.

Return Value

Void.

Syntax

object.**AutoPrint** hDC

Parameters

hDC

The HDC of the printer to be printed to.

Data Type: HDC

Param: IN

Default Value: None

Comments

This method simplifies the work of the developer needs to do to print a document. Sample Visual Basic code for printing an HTML document is as follows:

```
Printer.Print
html1.AutoPrint Printer.hDC
Printer.EndDoc
```

BeginPrinting HTML Method

Description

Sets up the printing of the displayed document.

Return Value

Void.

Syntax

object.**BeginPrinting** hDC [X] [Y] [Width] [Height] [DefaultHeader]
[DefaultTitle]

Parameters

hDC

The HDC of the printer to be printed to.

Data Type: HDC

Param: IN

Default Value: None

X

Optional. Used with the other optional parameters to define the rest of the area of the page to print to.

Data Type: Long

Param: IN

Default Value: None

Y

Optional. Used with the other optional parameters to define the rest of the area of the page to print to.

Data Type: Long

Param: IN

Default Value: None

Width

Optional. Used with the other optional parameters to define the rest of the area of the page to print to.

Data Type: Long

Param: IN

Default Value: None

Height

Optional. Used with the other optional parameters to define the rest of the area of the page to print to.

Data Type: Variant

Param: IN

Default Value: None

DefaultHeader

Optional. Placeholder. Not currently used.

Data Type: Variant

Param: IN

Default Value: None

DefaultTitle

Optional. Placeholder. Not currently used.

Data Type: BSTR

Param: IN

Default Value: None

Comments

This method sets up the HTML document for printing. The following is some sample Visual Basic code for printing an HTML document.

```
Dim page as Long
Dim DonePrinting as Boolean

Printer.Print
html1.BeginPrinting Printer.hDC
page=1
do
  html1.PrintPage Printer.hDC,page
  page=page+1
  DonePrinting=html1.IsPrintingDone(page)
  Printer.NewPage
Loop While (DonePrinting = False)
html1.EndPrinting
Printer.EndDoc
```

Cancel HTML Method

Description

Used to terminate document retrieval (including embedded documents), and optionally output a message at the end of the partially retrieved HTML page.

Return Value

Void.

Syntax

object.**Cancel** [*Message*]

Parameters

Message

Optional. Message to be appended to the HTML page.

Data Type: String

Param: IN

Default Value: None

Comments

If a message is specified, it will be enclosed in HTML tags, as shown here, and appended to the end of the page:

```
<HR><H2>Message</H2>
```

HTML tags are also allowed in the Message.

EndPrinting HTML Method

Description

Used to end the document print job.

Return Value

Void.

Syntax

object.**EndPrinting**

Comments

This method cleans up the changes made by BeginPrinting. The following is some sample Visual Basic code for printing an HTML document.

```
Dim page as Long
Dim DonePrinting as Boolean

Printer.Print
html1.BeginPrinting Printer.hDC
page=1
do
  html1.PrintPage Printer.hDC,page
  page=page+1
  DonePrinting=html1.IsPrintingDone(page)
  Printer.NewPage
Loop While (DonePrinting = False)
html1.EndPrinting
Printer.EndDoc
```

GetPlainText HTML Method

Description

Return the current text selection.

Return Value

BSTR.

Syntax

object.**GetPlainText** selected fancy

Parameters

selected

If selected is true, GetPlainText returns only the selected text; otherwise it returns all the text.

Data Type: Boolean

Param: IN

Default Value: N/A.

fancy

If fancy is true, the returned text uses a fancy format, i.e., horizontal lines are converted to dashed lines.

Data Type: Boolean

Param: IN

Default Value: N/A.

PrintPage HTML Method

Description

Used to print each page of the HTML document.

Return Value

void.

Syntax

object.**PrintPage** hDC pageNumber

Parameters

hDC

The printer HDC of the printer to print to.

Data Type: Long

Param: IN

Default Value: None.

pageNumber

The number of the page to be printed.

Data Type: Long

Param: IN

Default Value: None.

Comments

The following is some sample Visual Basic code for printing an HTML document.

```
Dim page as Long
Dim DonePrinting as Boolean

Printer.Print
html1.BeginPrinting Printer.hDC
page=1
do
  html1.PrintPage Printer.hDC,page
  page=page+1
  DonePrinting=html1.IsPrintingDone(page)
  Printer.NewPage
Loop While (DonePrinting = False)
html1.EndPrinting
Printer.EndDoc
```

RequestAllEmbedded HTML Method

Description

Requests retrieval of all embedded documents. If successful, the DoRequestEmbedded event will be activated.

Return Value

Void.

Syntax

object.**RequestAllEmbedded**

Parameters

None.

Comments

This method is used in conjunction with the DeferRetrieval property to control inline display of embedded documents.

RequestDoc HTML Method

Description

Requests retrieval of a new main document identified by the URL.

Return Value

Void.

Syntax

object.**RequestDoc** *URL*

Parameters

URL

Identifies the new main document to be retrieved.

Data Type: String

Param: IN

Default Value: http:

Comments

When RequestDoc is called, the DoRequestDoc event is activated and may be used to modify the DocStream to be used for retrieval, if desired. The RequestURL property will then be set to the URL parameter specified. The URL property will not be updated until retrieval is successfully underway and the BeginRetrieval event is activated.

SelectAll HTML Method

Description

Selects all the text in the current HTML view window.

Return Value

Void.

Syntax

object.**SelectAll**

HTML Events

The next series of Help topics describe the events activated by the HTML control. Each description includes the syntax, related parameters, their data type, default value, and whether the parameter is used for input or output (IN or OUT).

BeginRetrieval HTML Event

Description

This event is activated when document retrieval begins.

Syntax

*object*_**BeginRetrieval**

Parameters

None.

Comments

If the application uses a progress bar, it can be initialized at this time. The URL property will be copied from the RequestURL property immediately before the event is activated.

Click HTML Event

Description

This event is activated when the user presses and then releases the mouse button over an object.

Syntax

*object*_**Click**

Parameters

None.

DblClick HTML Event

Description

This event is activated when the user presses and releases the mouse button twice over an object.

Syntax

*object*_**DblClick**

Parameters

None.

DocInput HTML Event

Description

A DocInput related event that indicates the input data has been transferred or the DocInput state has changed.

Syntax

*object*_**DocInput** (*DocInput* **As DocInput**)

Parameters

DocInput

Object describing document input data for the current transfer.

Data Type: DocInput

Param: IN

Default Value: N/A

Comments

The DocInput event can be used in its basic form for notification of transfer progress, (for example, for updating a progress bar). The DocInput.BytesTotal, DocInput.BytesTransferred and DocInput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

To gain more power from this control, you can also use the DocInput event for data streaming. For more information, see DocInput Object Overview.

DocOutput HTML Event

Description

A DocOutput related event indicating that output data has been transferred or the DocOutput state has changed.

Syntax

*object*_**DocOutput** (*DocOutput* **As DocOutput**)

Parameters

DocOutput

Object describing document output data for the current transfer.

Data Type: DocOutput

Param: IN

Default Value: N/A

Comments

The DocOutput event can be used in its basic form to notify the user of transfer progress, (for example, for updating a progress bar). The DocOutput.BytesTotal, DocOutput.BytesTransferred and DocOutput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

To gain more power from this control, you can also use the DocOutput event for data streaming. For more information, see DocInput Object Overview.

DoNewElement HTML Event

Description

The event is activated during HTML parsing when a new element is added.

Syntax

*object*_**DoNewElement** (*ElemType* **As String**, *EndTag* **As Boolean**, *Attrs* **As HTMLAttrs**, *Text* **as String**, *EnableDefault* **As Boolean**)

Parameters

Elemtype

Element type for tags; empty string for character data.

Data Type: String

Param: IN

Default Value: None.

EndTag

True if an end tag; otherwise False.

Data Type: Boolean

Param: IN

Default Value: None.

Attrs

Collection of tag attributes, described in section HTMLAttrs Collection Properties of this chapter.

Data Type: HTMLAttrs

Param: IN

Default Value: None.

Text

Character data; empty for tags.

Data Type: String

Param: IN

Default Value: None.

EnableDefault

Overrides default processing. True indicates default processing, False means override defaults. If EnableDefault is set to false, the HTML control does not store data for this element, but continues parsing.

Data Type: Boolean

Param: IN/OUT

Default Value: True.

Comments

For character data, ElemType will be an empty string, and Text will contain the character data. For tags, the ElemType will contain the tag type, and the new element's attribute information can be retrieved using the HTMLAttrs collection argument.

DoRequestDoc HTML Event

Description

The event is activated when the user chooses a link to a different URL or when the RequestDoc method is called.

Syntax

*object*_**DoRequestDoc** (*URL* **As String**, *Element* **As HTMLElement**, *DocInput* **As DocInput**, *EnableDefault* **As Boolean**)

Parameters

URL

Identifies the requested document

Data Type: String

Param: IN

Default Value: None.

Element

Currently unused, but in the future it will identify the anchor element of the link selected by the user.

Data Type: HTMLElement

Param: IN

Default Value: None.

DocInput

May be modified to cause the control to accept input from another source.

Data Type: DocInput

Param: IN

Default Value: None.

EnableDefault

Overrides default processing. True indicates default processing, False means cancel default processing.

Data Type: Boolean

Param: IN/OUT

Default Value: True.

Default Action

The default action of DoRequestDoc depends on the URL type.

URL Type	Default Action
HTTP and File URL	DoRequestDoc creates a default DocInput object for retrieving the document
Other URL types or from a different source for any URL type	The DocInput property is set during event handling

Comments

If the event is not canceled, the RequestURL property will be set by the control. The URL property will not be updated until retrieval is successfully underway and the BeginRetrieval event is activated.

DoRequestEmbedded HTML Event

Description

The event is activated when an embedded document, such as an image is to be retrieved for inline display

Syntax

*object*_**DoRequestEmbedded** (*URL* **As String**, *Element* **As HTMLElement**, *DocInput* **As DocInput**, *EnableDefault* **As Boolean**)

Parameters

URL

Identifies the requested document

Data Type: String

Param: IN

Default Value: None.

Element

Currently unused, but in the future it will identify the HTML element of the embedded document.

Data Type: HTMLElement

Param: IN

Default Value: None.

DocInput

May be modified to cause the control to accept input from another source.

Data Type: DocInput

Param: IN

Default Value: None.

EnableDefault

Overrides default processing. True indicates default processing, False means cancel the request.

Data Type: Boolean

Param: IN/OUT

Default Value: True.

Default Action

The default action of DoRequestEmbedded depends on the URL type.

URL Type	**Default Action**
HTTP and File URL	DoRequestEmbedded creates a default DocInput object for retrieving the document.
Other URL types or from a different source for any URL type	The DocInput property is set during event handling.

{button ,JI(`NIA.HLP',`IDH_DocInput_Object_Properties')} **See Also**

DoRequestSubmit HTML Event

Description

The event is activated when the user selects form submission, or when the RequestSubmit method of the Form is called.

Syntax

*object*_**DoRequestSubmit** (*URL* **As String**, *Form* **As HTMLForm**, *DocOutput* **As DocOutput**, *EnableDefault* **As Boolean**)

Parameters

URL

Identifies the action URL for that form, and includes the search string for GET form methods as described in Comments.

Data Type: String

Param: IN

Default Value: None.

Form

Identifies the form being submitted, and is an item in the Forms collection

Data Type: HTMLForm

Param: IN

Default Value: None.

DocOutput

May be modified to cause output to another target.

Data Type: DocOutput

Param: IN

EnableDefault

Overrides default processing. True indicates default processing, False means override defaults. To cancel the submission request, set the EnableDefault parameter to False. If the event is not canceled, the RequestURL property will be set by the control. The URL property will not be updated until retrieval is successfully underway and the BeginRetrieval event is activated.

Data Type: Boolean

Param: IN/OUT

Default Value: True.

Default Action

The default action of DoRequestSubmit is to output the form's contents using HTTP, and input the reply as the next main document. To submit using a different source and/or target during event handling, you may modify the DocOutput property to specify some other target and link the DocInput property to receive the reply. To submit form data to another target without receiving the reply in the HTML control, modify the DocOutput property to some other target and unlink the DocInput property so that the reply document is discarded.

Comments

Currently, the form contents for submission always consist of URL-encoded field values contained in the Form.URLEncodedBody property. In the future, multipart content data may also be submitted for file uploading. If the form's submission method is GET (rather than POST), the string passed in the URL parameter of this event will have the URL-encoded body appended after the search character (question mark).

EndRetrieval HTML Event

Description

The event is activated when document retrieval, including embedded documents to be displayed inline, is complete.

Syntax

*object*_**EndRetrieval**

Parameters

None.

Comments

A progress bar could be terminated at this time.

Error HTML Event

Description

This event is activated when an error occurs in background processing (for example, failed to connect or failed to send or receive in the background).

Syntax

*object*_**Error** (*ErrCode* **As Integer**, *Description* **As String**, *Scode* **As Long**, *Source* **As String**, *HelpFile* **As String**, *HelpContext* **As Long**, *CancelDisplay* **As Boolean**)

Parameters

ErrCode

The short error code. For a list of possible HTML error codes see HTML Error Codes.

Description

String containing error information.

sCode

The long Scode.

Source

Error source.

HelpFile

Help file name.

HelpContext

Help file context.

CancelDisplay

Indicates whether to cancel the display. The default is TRUE (no display of the default error message box). If you do want to use the default message box, set CancelDisplay to FALSE.

GotFocus HTML Event

Description

The event is activated when an object receives the focus, either by user action, such as tabbing to or clicking the object, or by changing the focus in the code using the SetFocus method.

Syntax

*object*_**GotFocus**

Parameters

None.

KeyDown HTML Event

Description

The event is activated when the user presses a key while an object has the focus. All arguments of this event are standard.

Syntax

*object*_**KeyDown** (*KeyCode* **As Integer**, *Shift* **As Integer**)

Parameters

KeyCode

Indicates the key being pressed.

Data Type: Integer

Param: IN

Default Value: None.

Shift

Indicates whether a Shift, Ctrl, and/or Alt key was also pressed.

Data Type: Integer

Param: IN

Default Value: None.

KeyPress HTML Event

Description

The event is activated when the user presses and releases an ANSI key. All arguments of this event are standard.

Syntax

*object*_**KeyPress** (*KeyAsii* **As Integer**)

Parameters

KeyAsii

Indicates the ANSI key being pressed.

Data Type: Integer

Param: IN

Default Value: None.

KeyUp HTML Event

Description

The event is activated when the user releases a key while an object has the focus. All arguments of this event are standard.

Syntax

*object*_**KeyUp** (*KeyCode* **As Integer**, *Shift* **As Integer**)

Parameters

KeyCode

Indicates the key being released.

Data Type: Integer

Param: IN

Default Value: None.

Shift

Indicates whether a Shift, Ctrl, and/or Alt key was also pressed.

Data Type: Integer

Param: IN

Default Value: None.

LayoutComplete HTML Event

Description

The event is activated when layout of the HTML document is complete.

Syntax

*object*_**LayoutComplete**

Parameters

None.

Comments

Embedded document retrieval may not be complete, however, at least the size of each embedded document and the position of all elements has been determined.

LostFocus HTML Event

Description

The event is activated when an object loses the focus, either by user action, such as tabbing to or clicking the object, or by changing the focus in the code using the SetFocus method.

Syntax

*object*_**LostFocus**

Parameters

None.

MouseDown HTML Event

Description

The event is activated when the user presses a mouse button. All arguments of this event are standard.

Syntax

*object*_**MouseDown** (*Button* **As Integer**, *Shift* **As Integer,** *X* **As Float,** *Y* **As Float**)

Parameters

Button

Indicates which mouse button was pressed.

Data Type: Integer

Param: IN

Default Value: None.

Shift

Indicates whether a Shift, Ctrl, and/or Alt key was also pressed.

Data Type: Integer

Param: IN

Default Value: None.

X

Indicates the X axis of the pointer when the mouse button was pressed.

Data Type: Integer

Param: IN

Default Value: None.

Y

Indicates the Y axis of the pointer when the mouse button was pressed.

Data Type: Integer

Param: IN

Default Value: None.

MouseMove HTML Event

Description

The event is activated the user moves the mouse. All arguments of this event are standard.

Syntax

*object*_**MouseMove** (*Button* **As Integer**, *Shift* **As Integer,** *X* **As Float,** *Y* **As Float**)

Parameters

Button

Indicates the mouse button being moved.

Data Type: Integer

Param: IN

Default Value: None.

Shift

Indicates whether a Shift, Ctrl, and/or Alt key was also pressed.

Data Type: Integer

Param: IN

Default Value: None.

X

Indicates the X axis of the pointer when the mouse button was pressed.

Data Type: Integer

Param: IN

Default Value: None.

Y

Indicates the Y axis of the pointer when the mouse button was pressed.

Data Type: Integer

Param: IN

Default Value: None.

MouseUp HTML Event

Description

The event is activated when the user releases a mouse button. All arguments of this event are standard.

Syntax

*object*_**MouseUp** (*Button* **As Integer**, *Shift* **As Integer,** *X* **As Float,** *Y* **As Float**)

Parameters

Button

Indicates the mouse button being released.

Data Type: Integer

Param: IN

Default Value: None.

Shift

Indicates whether a Shift, Ctrl, and/or Alt key was also released.

Data Type: Integer

Param: IN

Default Value: None.

X

Indicates the X axis of the pointer when the mouse button was pressed.

Data Type: Integer

Param: IN

Default Value: None.

Y

Indicates the Y axis of the pointer when the mouse button was pressed.

Data Type: Integer

Param: IN

Default Value: None.

ParseComplete HTML Event

Description

The event is activated when parsing of the HTML document is complete.

Syntax

*object*_**ParseComplete**

Parameters

None.

Comments

Layout and embedded document retrieval may not be complete.

TimeOut HTML Event

Description

The event is activated after no data has been received within the time specified in the Timeout property.

Syntax

*object*_**TimeOut**

Parameters

None.

Comments

Although the Timeout value applies to all document retrieval, the Timeout event is activated only for the main document, not for embedded documents.

Event is an integer value that determines the type of Timeout event that will be enabled. Constants defined for enum types for events are:

Value	Meaning
prcConnectTimeout = 1	Timeout for connect. If connection is not established within the timeout period, the Timeout event will be activated.
prcReceiveTimeout = 2	Timeout for receiving data. If no data arrives within the timeout period, the Timeout event will be activated.
prcUserTimeout= 65	Timeout for user defined event. Use prcUserTimeout + [Integer] range for custom timeout events.

UpdateRetrieval HTML Event

Description

The event is activated periodically as the document and embedded objects are retrieved.

Syntax

*object*_**UpdateRetrieval**

Parameters

None.

Comments

The RetrieveBytesTotal and RetrieveBytesDone properties can be queried at the time this event is activated to update a progress bar.

HTMLAttrs Collection

An HTMLAttrs object is a collection containing HTMLAttr items. An HTMLAttrs collection is passed as an argument when the DoNewElement event is activated.

Count HTMLAttrs Collection Property

Description

The number of attributes in the collection.

Syntax

object.**Count**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

None.

Range

>=zero.

Item HTMLAttrs Collection Method

Description

Returns an item from the collection. The Item method is the default method for a collection.

Syntax

object.**Item** (*Index*)

Parameters

Index

Identifies the item in the collection. May be either an integer or a string. Integer indices identify an item by its one-based index. String indices identify an item by its Name property.

Data Type: Variant

Param: IN

Default Value: None.

HTMLAttr Item

An HTMLAttr object is an item in an HTMLAttrs collection. HTMLAttr items are used for specifying the attribute names and values of an HTML element.

HTMLAttr Item Properties

The HTMLAttr Items support the Name and Value properties.

Name HTMLAttr Item Property

Description

The attribute name. This string is never empty, and may be uppercase, lowercase or mixed case.

Syntax

HTMLAttrs.Item(*Index*).**Name**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

None.

Range

See RFC 1866 for attribute name syntax.

Value HTMLAttr Item Property

Description

The attribute value. This string may be empty. If not empty, the string is unescaped (decoded).

Syntax

HTMLAttrs.Item(*Index*).**Value**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

None.

Range

Any string.

HTMLForms Collection

An HTMLForms object is a collection containing HTMLForm items. The Forms property of the HTML control is an HTMLForms collection.

HTMLForms Collection Properties

The HTMLForms collection supports the Count property.

Count HTMLForms Collection Property

Description

The number of forms in the collection.

Syntax

object.**Count**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

None.

Range

>=zero.

HTMLForms Collection Methods

The HTMLForms collection supports the Item method.

Item HTMLForms Collection Method

Description

Returns an item from the collection. The Item method is the default method for a collection.

Syntax

object.**Item** (*Index*)

Parameters

Index

Identifies the item by its one-based index. Must be an integer.

Data Type: Variant

Param: IN

Default Value: None.

HTMLForm Item

An HTMLForm object is an item in an HTMLForms collection. HTMLForm items are used for submitting documents using HTTP.

Method HTMLForm Item Property

Description

The HTTP submission method for the form.

Syntax

object.**HTMLForms***(Index)*.**Method**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

None.

Range

“Get” or “Post”.

URL HTMLForm Item Property

Description

The action URL for the form.

Syntax

object.**HTMLForms***(Index)*.**URL**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

None.

Range

Valid URL.

URLEncodedBody HTMLForms Item Property

Description

The URL-encoded body text, representing the values of all form fields used for HTTP submission.

Syntax

object.**HTMLForms***(Index)*.**URLEncodedBody**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

None.

Range

URL-encoded string.

RequestSubmit HTMLForm Item Method

Description

Requests submission of a form.

Return Value

Void.

Syntax

object.**HTMLForms***(Index)*.**RequestSubmit**

Parameters

None.

Comments

When RequestSubmit is called, the DoRequestSubmit event is activated to determine the document target to be used for submission. The RequestURL property is then set to the action URL of the form. The URL property will not be updated until retrieval is successfully underway and the BeginRetrieval event is activated.

HTML Sample Sessions

Click on the buttons below to see examples of how the HTML control might be used in a real application.

{button ,JI(`NIA.HLP',`IDH_HTML_Session_1')} **Sample Session 1**
{button ,JI(`NIA.HLP',`IDH_HTML_Session_2')} **Sample Session 2**
{button ,JI(`NIA.HLP',`IDH_HTML_Session_3')} **Sample Session 3**

HTML Session 1

To point the HTML control to a particular URL, you can call the RequestDoc method, as shown here. This allows viewing (and then browsing from) a specific network document or local file.

```
    HTML1.RequestDoc("http://www.somecompany.com")
```

{button ,JI(`NIA.HLP',`IDH_HTML_Session_2')} **Sample Session 2**
{button ,JI(`NIA.HLP',`IDH_HTML_Session_3')} **Sample Session 3**

HTML Session 2

To change document retrieval behavior, you can override the DoRequestDoc event by setting the EnableDefault parameter to False during event handling. This will prevent automatic document retrieval when an HTML link is selected by the user, as shown here, and allow substituting different (or no) behavior. (The same technique can be used in handling the DoRequestEmbedded event to override retrieval of embedded documents such as inline images.)

```
Private Sub HTML1_DoRequestDoc(ByVal URL As String,
               ByVal Element As Variant,
               ByVal DocInput As DocInput,
               EnableDefault As Boolean)
  EnableDefault = False
  ‘now do something else with the URL parameter, such as:
  MsgBox(“URL requested: “ & URL)
End Sub
```

{button ,JI(`NIA.HLP',`IDH_HTML_Session_1')}   **Sample Session 1**
{button ,JI(`NIA.HLP',`IDH_HTML_Session_3')}   **Sample Session 3**

HTML Session 3

To use the HTML control as a nonvisual parser, set the Visible property to False at design time, and the EnableDefault parameter of the DoNewElement event to False during event handling. Setting the EnableDefault parameter to False will prevent storage and further processing of the element, eliminating overhead other than for parsing. This allows analyzing or processing the parsed HTML element information without the penalty of processing and storage overhead that would be needed for viewing and browsing.

```
Private Sub HTML1_DoNewElement(ByVal ElemType As String,
               ByVal EndTag As Boolean,
               ByVal Attrs As HTMLAttrs,
               ByVal Text As String,
               EnableDefault As Boolean)
  EnableDefault = False
  ‘now do something else with the parameters, such as:
  If ElemType = “” Then ‘is plain text data
        MsgBox(“Text: “ & Text)
  ElseIf EndTag Then
        MsgBox(“Element: </“ & ElemType &”>“)
  Else
        MsgBox(“Element: <“ & ElemType & “> with “ & Attrs.Count & “ attributes”)
  End If

End Sub
```

{button ,JI(`NIA.HLP',`IDH_HTML_Session_1')}   **Sample Session 1**
{button ,JI(`NIA.HLP',`IDH_HTML_Session_2')}   **Sample Session 2**

HTTP Client ActiveX Control Overview

The HTTP (Hypertext Transport Protocol) Control implements the HTTP Protocol Client based on the HTTP specification. This control lets you directly retrieve HTTP documents if no browsing or image processing is necessary.

It can be used by developers who implement HTML browsers or other services that need access to HTTP. For example, the HTML Control internally instantiates this object and uses it for HTTP transactions.

The HTTP Control uses a number of methods to retrieve or send (post) a document. It can retrieve MIME information about the document from the Headers collection property.

Properties, methods and events supported by the HTTP Client Control are summarized alphabetically in the following table.

Property	Method	Event
Blocking	AboutBox	Busy
BlockResult	Cancel	Cancel
Busy	GetDoc	DocInput
DocInput	SendDoc	DocOutput
DocOutput		Error
Document		Log
EnableTimer		ProtocolStateChanged
Errors		StateChanged
LocalPort		TimeOut
Logging		
Method		
NotificationMode		
ProtocolState		
ProtocolStateString		
RemoteHost		
RemotePort		
ReplyCode		
ReplyString		
SleepTime		
SocketHandle		
Timeout		
URL		
HTTP Proxy Object		

Using the HTTP Control

To use the HTTP ActiveX Control you must choose the HTTP toolbox icon.

There should be no speed overhead and response delay other than the one given by the network. This control uses and is dependent on the DocStream objects (DocInput and DocOutput). For more information, see DocStream.

Blocking HTTP Client Property

Description

Indicates whether methods should block until complete or not.

Syntax

object.**Blocking** *[=Boolean]*

Permission

W (Read/Write)

Availability

D (Design) and R (Runtime)

Data Type

Boolean.

Default Value

False.

Range

True or False.

Comments

If Blocking is set to True, then a method will not be completed until the proper event(s) for a given request are fired, if applicable.

BlockResult HTTP Client Property

Description

Returns the result value of the last blocking method called.

Syntax

object.**BlockResult**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

BlockingResultConstants.

Default Value

icBlockOK..

Range

Name	Value	Description
icBlockOK	0	Blocking method was successful.
IcTimedOut	1	Blocking method returned due to timeout.
IcErrorExit	2	Blocking method returned due to an error.
IcBlockCancel	3	Blocking method returned due to cancel.
IcUserQuit	4	Blocking method returned due application end.

Busy HTTP Client Property

Description

Indicates if a command is in progress.

Syntax

object.**Busy** *[= Boolean]*

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Boolean.

Default Value

N/A.

Range

True or False

DocInput HTTP Client Property

Description

Object describing input information for the document being transferred.

Syntax

object.**DocInput**

Permission

R (Read-only).

Availability

R (Runtime)

Data Type

DocInput.

Default Value

N/A.

Range

N/A.

Comments

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

Properties of the DocInput object may be set before calling the SendDoc method or they may be passed as arguments to this method. The DocInput object is also used for conveying information about the progress of the document transfer and for data linking and streaming.

For more information, see DocInput event and Common Control Objects.

DocOutput HTTP Client Property

Description

Object describing output information for the document being transferred.

Syntax

object.**DocOutput**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

DocOutput.

Default Value

N/A.

Range

N/A.

Comments

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

Properties of the DocOutput object may be set before calling the GetDoc method or they may be passed as arguments to this method. The DocOutput object is also used for conveying information about the progress of the document transfer, and for data linking and streaming.

For more information, see DocOutput event and Common Control Objects.

Document HTTP Client Property

Description

Identifies the target document. The Document property can be used with RemoteHost to identify the URL. It can also be used instead of the URL.

Syntax

object.**Document** [= *String*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

String.

Default Value

Empty String.

Range

N/A.

EnableTimer HTTP Client Property

Description

Enable timer for the specified event.

Syntax

object.**EnableTimer (***event***)** [= *Boolean*]

Permission

W (Write Only).

Note: This is the only control property that is Write only.

Availability

R (Runtime)

Data Type

Boolean.

Default Value

False. (The timer for this event will not be enabled.)

Range

True or False

Comments

Event is an integer value that determines the type of Timeout event that will be enabled. Constants defined for enum types for events are:

Value	Meaning
prcConnectTimeout = 1	Timeout for connect. If connection is not established within the timeout period, the Timeout event will be activated.
prcReceiveTimeout = 2	Timeout for receiving data. If no data arrives within the timeout period, the Timeout event will be activated.
prcUserTimeout= 65	Timeout for user defined event. Use prcUserTimeout + [Integer] range for custom timeout events.

Errors HTTP Client Property

Description

A collection of errors that can be accessed for details about the last error that occurred. This collection should be used within an Error event if information passed through the Error event is not sufficient. For more details, see icError Item Overview.

Syntax

object.**Errors**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

icErrors.

Default Value

N/A.

Range

N/A.

LocalPort HTTP Client Property

Description

Designates the local port to use.

Syntax

object.**LocalPort** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

0.

Range

0 - 65535

Logging HTTP Client Property

Description

Indicates whether log events should be fired when log data is available.

Syntax

object.**Logging** [= *Boolean*]

Permission

W (Read/Write).

Availability

D (Design) and R(Runtime).

Data Type

Boolean.

Default Value

False.

Range

True or False

Method HTTP Client Property

Description

Method used to retrieve or post (send) the document.

Syntax

object.**Method** [= *Integer*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

Integer.

Default Value

1 (prcGet)

Range

1-4. The possible values are:

Value	Meaning
prcGet = 1	Get method requests the whole document.
prcHead = 2	Head method requests only the headers of a document.
prcPost = 3	Post method posts the whole document to the server as a sub-ordinate of the document specified by the URL.
prcPut = 4	Put method puts the whole document to the server. The document replaces an existing document specified by the URL.

NotificationMode HTTP Client Property

Description

Determines when notification is issued for incoming data. Notification can also be suspended.

Syntax

object.**NotificationMode** [= *Integer*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

NotificationModeConstants.

Default Value

icContinuousMode.

Range

0-maximum unsigned long. At present, the values are:

Name	Value	Description
icCompleteMode	0	icCompleteMode: notification is provided

		when there is a complete response.
icContinuousMode	1	icContinuousMode: an event is repeatedly activated when new data arrives from the connection.

ProtocolState HTTP Client Property

Description

This property specifies the current state of the protocol.

Syntax

object.**ProtocolState**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Integer.

Default Value

prcBase.

Range

0-1. Constants defined for the enum types of ProtocolState property are:

Value	Meaning
httpBase = 0	Base state before connection to server is established.
httpTranferring = 1	Data is being transferred.

ProtocolStateString HTTP Client Property

Description

String representation of ProtocolState.

Syntax

object.**ProtocolStateString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

“Base”.

Range

N/A.

RemoteHost HTTP Client Property

Description

The remote machine to connect to if the RemoteHost parameter in the Connect method is missing. You can either provide a host name or an IP address string in dotted format. For example, “127.0.0.1”.

Note: This is the default property of the control.

Syntax

object.**RemoteHost** [= *String*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

String.

Default Value

“127.0.0.1”.

Range

N/A.

RemotePort HTTP Client Property

Description

The remote port number to which to connect.

Syntax

object.**RemotePort** [= *Long*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

Long.

Default Value

80.

Range

1-65535.

ReplyCode HTTP Client Property

Description

The value of the reply code is a protocol specific number that determines the result of the last request, as returned in the ReplyString property.

Syntax

object.**ReplyCode**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

0.

Range

0-(undefined)

ReplyString HTTP Client Property

Description

Lists the last reply string sent by the HTTP Server to the client as a result of a request.

Syntax

object.**ReplyString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

SleepTime HTTP Client Property

Description

Specifies the sleep time between checking messages, if Blocking is True.

Syntax

object.**SleepTime** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

10 ms.

Range

>=zero.

Comments

Only applies when in Blocking mode.

SocketHandle HTTP Client Property

Description

Socket handle for the primary connection (Request/Reply connection).

Syntax

object.**SocketHandle**

Permission

R (Read only)

Availability

R (Runtime).

Data Type

Long.

Default Value

N/A.

Range

>=0

Comments

Some protocols require more than one connection. SocketHandle is the handle for the request/reply connection.

State HTTP Client Property

Description

This property specifies the connection state of the control.

Syntax

object.**State**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Integer.

Default Value

prcDisconnected.

Range

1-6. Constants defined for enum types of State property are:

Value	Meaning
prcConnecting = 1	Connecting. Connect has been requested, waiting for connect acknowledge.
prcResolvingHost = 2	Resolving Host. Occurs only when RemoteHost is in name format rather than dot-delimited IP format.
prcHostResolved = 3	Resolved the host. Occurs only if ResolvingHost state has been entered previously.
prcConnected = 4	Connection established.
prcDisconnecting = 5	Connection closed. Disconnect has been initiated.
prcDisconnected = 6	Initial state when protocol object is instantiated, before Connect has been initiated, after a Connect attempt failed or after Disconnect performed.

StateString HTTP Client Property

Description

A string representation of State.

Syntax

object.**StateString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

"Disconnected".

Range

N/A.

Timeout HTTP Client Property

Description

Timeout value for the specified event. The event is specified by entering:

```
Timeout(short event)
```

Syntax

object.**Timeout** (*event*) [= *Long*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

Long.

Default Value

0.

Range

0-maximum unsigned long. Constants defined for enum types for events are:

Value	Meaning
prcConnectTimeout = 1	Timeout for connect. If connection is not established within the timeout period, the Timeout event will be activated.
prcReceiveTimeout = 2	Timeout for receiving data. If no data arrives within the timeout period, the Timeout event will be activated.
prcUserTimeout= 65	Timeout for user defined event. Use prcUserTimeout + [Integer] range for custom timeout events.

URL HTTP Client Property

Description

URL string identifying the current document being transferred. URL format is:

```
HTTP://host:port/documentnameandpath
```

Syntax

object.**URL** [= S*tring*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

String.

Default Value

Empty string.

Range

Valid URL.

Comments

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.

AboutBox HTTP Client Method

Description

Shows information about this control.

Return Value

Void.

Syntax

object.**AboutBox**

Parameters

None.

Cancel HTTP Client Method

Description

Cancels a pending request.

Return Value

Void.

Syntax

object.**Cancel**

Parameters

None.

GetDoc HTTP Client Method

Description

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

Return Value

Void.

Syntax

object.**GetDoc** [*URL*], [*Headers*], [*OutputFile*]

Parameters

URL

Optional. The URL identifying the remote document to be retrieved.

Data Type: String

Param: IN

Default Value: DocInput.URL

Headers

Optional. Headers used for requesting the document. This argument only applies to protocols where request headers can be specified (for example, HTTP).

Data Type: DocHeaders

Param: IN

Default Value: DocInput.Headers

OutputFile

Optional. A local file to which the retrieved document will be written.

Data Type: String

Param: IN

Default Value: DocOutput.Filename

Comments

The GetDoc method permits retrieving a document 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) may be omitted and will default to the correct type for this control. For example, when using the HTTP control, the "http:" string may 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.

For more information, see DocInput and DocOutput Objects and Common Control Objects.

SendDoc HTTP Client Method

Description

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

Return Value

Void.

Syntax

object.**SendDoc** [*URL*], [*Headers*], [*InputData*], [*InputFile*], [*OutputFile*]

Parameters

URL

Optional. The URL identifying the remote document to be sent. If specified, the URL property will be set to this value.

Data Type: String

Param: IN

Default Value: DocInput.URL

Headers

Optional. Headers used for sending the document. This argument only applies to protocols where document headers can be sent (for example, SMTP and HTTP).

Data Type: DocHeaders

Param: IN

Default Value: DocInput.Headers

InputData

Optional. A data buffer containing the document to be sent.

Data Type: VARIANT

Param: IN

Default Value: DocInput.SetData

InputFile

Optional. A local file containing the document to be sent.

Data Type: String

Param: IN

Default Value: DocInput.Filename

OutputFile

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).

Data Type: String

Param: IN

Default Value: DocOutput.Filename

Comments

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) may be omitted and will default to the correct type for this control. For example, when using the HTTP control, the "http:" string may 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.

For more information see DocInput Object Overview and DocOutput Object Overview.

HTTP Client Events

Events are used for HTTP client notification. They indicate that an action has been requested and processed. Any errors which occur during command processing result in the Error event being called with appropriate error codes. Error codes, state changes, and protocol return values are usually checked during event processing.

The following sections describe the events supported by the HTTP Client Control. Each event description includes the syntax, related parameters, their data type, default value, and whether the parameter is used for input or output (IN or OUT).

Busy HTTP Client Event

Description

This event is activated when a command is in progress or when a command has completed.

Syntax

*object*_**Busy** (*Busy* **As Boolean**)

Parameters

Busy

Indicates whether or not a command is in progress.

Data Type: Boolean. If the argument is True, a command is in progress.

Cancel HTTP Client Event

Description

This event is activated after a cancellation request has been completed and satisfied. After this event the object's state changes to prcDisconnected.

Syntax

*object*_**Cancel**

Parameters

None.

DocInput HTTP Client Event

Description

A DocInput related event that indicates the input data has been transferred or the DocInput state has changed.

Syntax

*object*_**DocInput** (*DocInput* **As DocInput**)

Parameters

DocInput

Object describing document input data for the current transfer.

Data Type: DocInput

Comments

The DocInput event can be used in its basic form for notification of transfer progress, (for example, for updating a progress bar). The DocInput.BytesTotal, DocInput.BytesTransferred and DocInput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

To gain more power from this control, you can also use the DocInput event for data streaming. For more information, see For more information see DocInput Object Overview.

DocOutput HTTP Client Event

Description

A DocOutput related event indicating that output data has been transferred.

Syntax

*object*_**DocOutput** (*DocOutput* **As DocOutput**)

Parameters

DocOutput

Object describing document output data for the current transfer.

Data Type: DocOutput

Comments

The DocOutput event can be used in its basic form to notify the user of transfer progress, (for example, for updating a progress bar). The DocOutput.BytesTotal, DocOutput.BytesTransferred and DocOutput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

To gain more power from this control, you can also use the DocOutput event for data streaming. For more information, see For more information see DocInput Object Overview.

Error HTTP Client Event

Description

This event is activated when an error occurs in background processing (for example, failed to connect or failed to send or receive in the background).

Syntax

*object*_**Error** (*ErrCode* **As Integer**, *Description* **As String**, *Scode* **As Long**, *Source* **As String**, *HelpFile* **As String**, *HelpContext* **As Long**, *CancelDisplay* **As Boolean**)

Parameters

ErrCode

The short error code.

Description

String containing error information.

sCode

The long Scode.

Source

Error source.

HelpFile

Help file name.

HelpContext

Help file context.

CancelDisplay

Indicates whether to cancel the display. The default is TRUE (no display of the default error message box). If you do want to use the default message box, set CancelDisplay to FALSE.

Log HTTP Client Event

Description

This event is fired when logging data is available.

Syntax

*object*_**Log**

Parameters

None.

ProtocolStateChanged HTTP Client Event

Description

This event is activated whenever the protocol state changes.

Syntax

*object*_**ProtocolStateChanged** (*State* **As Integer**)

Parameters

Refer to the ProtocolState property and ProtocolStateString for possible values of the state parameter.

StateChanged HTTP Client Event

Description

This event is activated whenever the state of the transport state changes.

Syntax

*object*_**StateChanged** (*State* **As Integer**)

Parameters

Refer to the State property and StateString for possible values of the state parameter.

TimeOut HTTP Client Event

Description

This event is activated when the timer for the specified event expires.

Syntax

*object*_**TimeOut** (ByVal *Event* **As Integer,** *Continue* **As Boolean**)

Parameters

Event

Defines the event to which the time interval applies.

Data Type: Short

Continue

Determines if the timer is active or not. Set *Continue* to TRUE to keep the timer active.

Data Type: Boolean

Default Value: False

Comments

Event is an integer value that determines the type of Timeout event that will be enabled. Constants defined for enum types for events are:

Value	**Meaning**
prcConnectTimeout = 1	Timeout for connect. If connection is not established within the timeout period, the Timeout event will be activated.
prcReceiveTimeout = 2	Timeout for receiving data. If no data arrives within the timeout period, the Timeout event will be activated.
prcUserTimeout= 65	Timeout for user defined event. Use prcUserTimeout + [Integer] range for custom timeout events.

{button ,JI(`NIA.HLP',`IDH_Timeout')} **See Also**

HTTP Localization

Error strings for this control are localized.

HTTP Sample Sessions

Click on one of the items below to see an example of how to use HTTP Control to receive an HTTP document, parse it, and process the contents of the document.

- Receiving a Reply
- Sending a Request Using DocStream
- Processing Errors

Receiving a Reply

To receive a reply and process the contents of the document, you might code:

```
Private Sub HTTPCT1_DocOutput(ByVal docOutput As docOutput)

If (docOutput.State = 3) Then 'DATA
contentType = docOutput.DocHeaders.Item("content-type").Value
Dim data As Variant

  if InStr$(1, contentType, “text/”, 1) then
        ‘ retrieve text data
        docOutput.GetData data
        txtOutput.Text = txtOutput.Text & data
        txtLog = txtLog & “received html data of length = “ & str$(Len(data)) &
NL
  else
    ‘ Image or other non-html data
  docOutput.GetData(data, vbArray + vbByte)
  txtLog = txtLog & “received binary data of length = “ & Str$(UBound(data)) & NL
  end if
End If
End Sub
```

See Also

Sending a Request Using DocStream
Processing Errors

Sending a Request Using DocStream

The following code shows how to send a request using DocStream objects.

```
Private Sub Command1_Click()
‘ Send request
‘ use the following line if you want to write the data into a file, in which
case
‘ no processing of DocOutput event will be necessary
‘ HTTTCT1.DocOutput.FileName = “file.tmp”
HTTPCT1.GetDoc "http://www.netmanage.com/"
End Sub
```

See Also

Receiving a Reply
Processing Errors

Processing Errors

The following code can be used to process any errors that might occur.

```
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)
CancelDisplay = True
errLog = errLog & "Error " & Str$(Number) & ": " & Description & NL
End Sub
```

See Also

Receiving a Reply
Sending a Request Using DocStream

HTTP Proxy Object

The Proxy object provides an interface to get and set information used to access hosts external to a firewall protected domain. Through this interface, you can set a few parameters and then access external hosts as you would do if the firewall did not exist.

A HTTP Proxy server will process URLs and pass back the obtained information.

The HTTP Proxy properties are as follows:

- Host
- Mode
- Port

Host HTTP Proxy Property

Description

The address or name of the remote machine that is the proxy server. You can either provide a host name or an IP address string in dotted format. For example, 127.0.0.1.

Syntax

object.**Proxy.Host** [= String]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

BSTR.

Default Value

“127.0.0.1”

Range

N/A.

Mode HTTP Proxy Property

Description

Determines whether the connection is to be made to the proxy server or to the HTTP.RemoteHost server.

Syntax

object.**Proxy.Mode** [= Boolean]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

Boolean.

Default Value

FALSE.

Range

True or False.

Constant	**Meaning**
False	Proxy service is not used.
True	Connections are made to the proxy server and the URL is passed to that server for processing

Port HTTP Proxy Property

Description

The port number of the proxy server host to connect to.

Syntax

object.**Proxy.Port** [= Long]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

Long.

Default Value

80.

Range

1-65535.

Comment

Although the data type is long, WinSock conventions limit the maximum port   number to be the maximum signed short value.

HTTP Server ActiveX Control Overview

The HTTP (Hypertext Transport Protocol) Server Control implements the HTTP Server based on the HTTP specification. This control enables users to write HTTP servers, add security checking/filtering, gather statistics, and other functions.

The HTTP Server keeps a collection of connected sessions. For each session, when there is a request coming in, the Request event is fired. The Method property on the session shows what kind of request it is and the Document property is the requested document. You can decide what to do with the request.

The following is a list of possible request methods.

- GET
- HEAD
- PUT
- POST

Properties, methods and events supported by the HTTP Client Control are summarized alphabetically in the following table.

Property	**Method**	**Event**
Server		
Blocking	CloseAll	Accept
BlockResult	AboutBox	CloseAll
DefaultDocument	Start	DocInput
Errors	Stop	DocOutput
ListenPort		Error
LocalPort		RemoteHostName
MaxConnections		Request
RootDirectory		StateChanged
		TimeOut
SleepTime		
SocketHandle		
Sessions Collection		
Count	Item	
Session Item		
DocInput	Close	
DocOutput	ReplyDoc	
NotificationMode	SendData	
RemoteHostIP		
RemoteHostName		
RemotePort		
ReplyString		
RequestString		
State		
StateString		
Tag		
ThreadID		
Timeout		

Blocking HTTP Server Property

Description

Indicates whether methods should block until complete or not.

Syntax

object.**Blocking** *[=Boolean]*

Permission

W (Read/Write)

Availability

D (Design) and R (Runtime)

Data Type

Boolean.

Default Value

False.

Range

True or False.

Comments

If Blocking is set to True, then a method will not be completed until the proper event(s) for a given request are fired, if applicable.

BlockResult HTTP Server Property

Description

Returns the result value of the last blocking method called.

Syntax

object.**BlockResult**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

BlockingResultConstants.

Default Value

icBlockOK..

Range

Name	Value	Description
icBlockOK	0	Blocking method was successful.
IcTimedOut	1	Blocking method returned due to timeout.
IcErrorExit	2	Blocking method returned due to an error.
IcBlockCancel	3	Blocking method returned due to cancel.
IcUserQuit	4	Blocking method returned due application end.

DefaultDocument HTTP Server Property

Description

Name of the default document for default hanlding of request.

Syntax

object.**DefaultDocument** = *index.htm*

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

BSTR.

Default Value

“index.htm”.

Range

N/A.

Errors HTTP Server Property

Description

A collection of errors that can be accessed for details about the last error that occurred. This collection should be used within an Error event if information passed through the Error event is not sufficient. For more details, see icErrors Item Overview.

Syntax

object.**Errors**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

icErrors.

Default Value

N/A.

Range

N/A.

LocalPort HTTP Server Property

Description

Designates the local port to use.

Syntax

object.**LocalPort** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

0.

Range

0 - 65535

ListenPort HTTP Server Property

Description

The port number to listen on.

Syntax

object.**ListenPort** [= Long]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default

0.

Range

0-65535.

MaxConnections HTTP Server Property

Description

The maximum number of clients allowed to connect to the server. If connecting clients exceeds MaxConnections, the Error event is fired and new connections are rejected until the maximum number of connections drops below the MaxConnection limit.

Syntax

object.**MaxConnections** [=*lLong*]

Permission

W (Read/Write).

Availability

D (Design).

Data Type

Long.

Default Value

100.

Range

0-65535

RootDirectory HTTP Server Property

Description

Root directory for default handling of requests.

Syntax

object.**RootDirectory** = [string]

Permission

W (Read/Write).

Availability

D (Design).

Data Type

BSTR.

Default Value

c:\netmanag.32\webdocs.

Range

N/A.

Sessions HTTP Server Property

Description

The server session collection object. Session object provides access to a connection accepted from a client. Sessions can be enumerated and actions can be performed on them. Normally a session object is passed as an argument to an event and the action can be performed on the object.

Syntax

object.**Session** [= *Object*]

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Object.

Default Value

N/A.

Range

N/A.

SleepTime HTTP Server Property

Description

Specifies the sleep time between checking messages, if Blocking is True.

Syntax

object.**SleepTime** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

10 ms.

Range

>=zero.

Comments

Only applies when in Blocking mode.

SocketHandle HTTP Server Property

Description

Socket handle for the primary connection (Request/Reply connection).

Syntax

object.**SocketHandle**

Permission

R (Read only)

Availability

R (Runtime).

Data Type

Long.

Default Value

N/A.

Range

>=0

Comments

Some protocols require more than one connection. SocketHandle is the handle for the request/reply connection.

Count Sessions Collection Property

Description

The number of items in the collection.

Syntax

object.**Count** [= *Integer*]

Permission

R (Read-only).

Availability

R(Runtime).

Data Type

Long.

Default Value

N/A.

Range

0-65535

DocInput Session Item Property (HTTP)

Description

Object describing input information for the document being transferred.

Syntax

object.**DocInput**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

DocInput.

Default Value

N/A.

Range

N/A.

Comments

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

Properties of the DocInput object may be set before calling the SendDoc method or they may be passed as arguments to this method. The DocInput object is also used for conveying information about the progress of the document transfer and for data linking and streaming.

For more information, see DocInput event and Common Control Objects.

DocOutput Session Item Property (HTTP)

Description

Object describing output information for the document being transferred.

Syntax

object.**DocOutput**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

DocOutput.

Default Value

N/A.

Range

N/A.

Comments

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

Properties of the DocOutput object may be set before calling the GetDoc method or they may be passed as arguments to this method. The DocOutput object is also used for conveying information about the progress of the document transfer, and for data linking and streaming.

For more information, see DocOutput Event and Common Control Objects.

Document Session Item Property

Description

Identifies the target document in the client request.

Syntax

object.**Document** = [string]

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

BSTR.

Default Value

N/A.

Range

N/A.

Method Session Item Property

Description

HTTP request type.

Syntax

object.**Method** = [*Integer*]

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

MethodConstants.

Default Value

N/A.

Range

1-4. The possible values are:

Value	Meaning
prcGet = 1	Get method request the whole document.
prcHead = 2	Head method requests only the headers of a document.
prcPost = 3	Post method posts the whole document to the server.
prcPut = 4	Put method puts the whole document to the server.

NotificationMode Session Item Property

Description

Determines when notification is issued for incoming data. Notification can also be suspended.

Syntax

object.**NotificationMode** [= *Integer*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

NotificationModeConstants.

Default Value

1.

Range

0-maximum unsigned long. At present, the values are:

Name	Value	Description
icCompleteMode	0	COMPLETE: notification is provided when there is a complete response.

icContinueousMode	1	CONTINUOUS: an event is repeatedly activated when new data arrives from the connection.

RemoteHostIP Session Item Property

Description

Remote host IP address string for the connected session. Can be used by the server to display information about all sessions.

Syntax

object.**RemoteHostIP** [= *String*]

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

BSTR.

Default Value

N/A

Range

N/A.

RemoteHostName Session Item Property

Description

Remote host's official name for the connected session. Can be used by the server to display information about all sessions. This property can be examined after the RemoteHostName event is successfully fired. See *RemoteHostName* event for additional information.

Syntax

object.**RemoteHostName** [= *String*]

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

BSTR.

Default Value

N/A

Range

N/A.

RemotePort Session Item Property (HTTP)

Description

The remote port number to which to connect.

Syntax

object.**RemotePort** [= *Long*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

Long.

Default Value

8889

Range

1-65535.

ReplyString Session Item Property (HTTP)

Description

Buffer contains an ASCII reply string for the session.

Syntax

object.**ReplyString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

State Session Item Property

Description

This property specifies the connection state of the control.

Syntax

object.**State**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Integer.

Default Value

prcDisconnected.

Range

1-6. Constants defined for enum types of State property are:

Value	**Meaning**
prcConnecting = 1	Connecting. Connect has been requested, waiting for connect acknowledge.
prcResolvingHost = 2	Resolving Host. Occurs when RemoteHost is in name format rather than dot-delimited IP format.
prcHostResolved = 3	Resolved the host. Occurs only if ResolvingHost state has been entered previously.
prcConnected = 4	Connection established.
prcDisconnecting = 5	Connection closed. Disconnect has been initiated.
prcDisconnected = 6	Initial state when protocol object is instantiated, before Connect has been initiated, after a Connect attempt failed or after Disconnect performed.

StateString Session Item Property (HTTP)

Description

A string representation of State.

Syntax

object.**StateString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

"Disconnected".

Range

N/A.

Tag Session Item Property

Description

A placeholder for user data.

Syntax

object.**Tag**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Variant.

Default Value

Empty.

Range

N/A.

ThreadID Session Item Property (HTTP)

Description

Sessions thread identifier

Syntax

object.**ThreadID**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

N/A.

Range

N/A.

Timeout Session Item Property (HTTP)

Description

The time in seconds. Timeout event is fired when a connected session has no activity (no request from a client). Normally, the client is prevented from timing out when a data transfer is taking place. When the Timeout event is fired, user may choose to log the client out by calling the Close method on the session. If the user did not choose to log the client out, the Timeout event will be fired again when there is still no activity within the Timeout period.

Syntax

object.**Timeout** (*event*) [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

300 seconds (5 minutes).

Range

0-65535

HTTP Server Methods

Methods are called to perform a particular operation. The next series of Help topics describe the methods performed by the HTTP Server Control.

AboutBox HTTP Server Method

Description

Shows information about this control.

Return Value

Void.

Syntax

object.**AboutBox**

Parameters

None.

CloseAll HTTP Server Method

Description

Close all connected sessions. The CloseAll event is fired.

Return Value

Void.

Syntax

object.**CloseAll**

Parameters

None

Start HTTP Server Method

Description

Opens the server up for listening on the port specified by the ListenPort property.

Return Value

Void.

Syntax

object.**Start**

Parameters

None

Stop HTTP Server Method

Description

Server stops listening for incoming connections.

Return Value

Void.

Syntax

object.**Stop** [*DisconnectSession*]

Parameters

DisconnectSession

Optional. If DisConnectSession is true, all connected sessions are disconnected. If DisconnectSessions is false, connected sessions will not be closed.

Data Type: Boolean

Param: IN

Default Value: True

Item Sessions Collection Method

Description

Returns the session object from the collection. This method is the default method for a collection.

Return Value

Session object.

Syntax

object.**Item** index

Parameters

index

Index is an integer number.

Data Type: Variant

Param: IN

Default Value: None

Close Session Item Method

Description

Close the session by disconnecting. Error event is fired in case of failure. StateChanged event is fired after the connection is closed.

Return Value

Void.

Syntax

object.**Close**

Parameters

None

ReplyDoc Session Item Method (HTTP)

Description

Initiates sending of a document

Return Value

Void.

Syntax

object.**ReplyDoc** [*Header*] [*InputData*] [*InputFiler*]

Parameters

Header

Optional. Header used for sending the document. This argument only applies for protocols where document headers can be sent.

Data Type: Docheaders.

Param: IN

Default Value: DocInput, Headers

Inputdata

Optional. A data buffer containing the document to be sent

Data Type: VARIANT.

Param: IN

Default Value: DocInput, SetData

InputFile

Optional. A local file containing the document to be sent.

Data Type: BSTR.

Param: IN

Default Value: DocInput, FileName

Comments

The ReplyDoc method allows sending (posting or putting) a document.

For basic use of this method, 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 properties of the DocInput and DocOutput objects can be set before calling ReplyDoc. The DocInput and DocOutput events can also be used for transfering data using streaming rather than local files. See the DocInput and DocOutput properties, the DocInput and DocOutput events, and the separate DocInput and DocOutput object documentation for more information.

SendData Session Item Method (HTTP)

Description

Send small amount of data to client. Large amount of data should be sent using DocStream. (See DocInput and ReplyDoc in this Chapter.)

Return Value

Void.

Syntax

object.**SendData** *data*

Parameters

Data

Data to be sent

Data Type: VARIANT.

Param: IN

Default Value: N/A

HTTP Server Events

Events are used for protocol server notification. They indicate that an action has been requested and processed. Any errors which occur during command processing result in the Error event being called with appropriate error codes. Error codes, state changes, and protocol return values are usually checked during event processing.

The Help topics that follow describe the events supported by the HTTP Server Control. Each description includes the syntax, related parameters, their data type, default value, and whether the parameter is used for input or output (IN or OUT). For a complete listing of HTTP Server Events, see HTTP Server ActiveX Control Overview.

Accept HTTP Server Event

Description

This event is fired when there is an incoming connection request.

Syntax

*object*_**Accept** (*Session* **As Object,** *AcceptConnection* **As Boolean**)

Parameters

Session

Session's property RemoteHostIP and RemotePort can be examined and used to decide whether to accept the session or not.

AcceptConnection

To reject the connection, *AcceptConnection* needs to be set to FALSE. The default value for *AcceptConnection* is TRUE

CloseAll HTTP Server Event

Description

This event is fired after all sessions have been closed.

Syntax

*object*_**CloseAll**

Parameters

None

DocInput HTTP Server Event

Description

A DocInput related event that indicates the input data has been transferred or the DocInput state has changed.

Syntax

*object*_**DocInput** (*Session* **As Object**, *DocInput* **As DocInput**)

Parameters

Session

The session object on which transfer of input data happens.

Data Type: Object

Default Value: N/A

DocInput

Object describing document input data for the current transfer.

Data Type: DocInput

Default Value: N/A

Comments

The DocInput event can be used in its basic form for notification of transfer progress, (for example, for updating a progress bar). The DocInput.BytesTotal, DocInput.BytesTransferred and DocInput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

To gain more power from this control, you can also use the DocInput event for data streaming.

DoctInput and DocOutput events are also fired when the DocInput/DocOutput states have changed.

DocOutput HTTP Server Event

Description

A DocOutput related event indicating that output data has been transferred or the DocOutput state has changed

Syntax

*object*_**DocOutput** (*Session* **As Object**, *DocOutput* **As DocOutput**)

Parameters

Session

The object on which output data transfer happens.

Data Type: Object

Default Value: N/A

DocOutput

Object describing document output data for the current transfer.

Data Type: DocOutput

Default Value: N/A

Comments

The DocOutput event can be used in its basic form to notify the user of transfer progress, (for example, for updating a progress bar). The DocOutput.BytesTotal, DocOutput.BytesTransferred and DocOutput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

To gain more power from this control, you can also use the DocOutput event for data streaming.

DoctInput and DocOutput events are also fired when the DocInput/DocOutput states have changed.

Error HTTP Server Event

Description

This event is activated when an error occurs in background processing (for example, failed to connect or failed to send or receive in the background).

Syntax

*object*_**Error** (*ErrCode* **As Integer**, *Description* **As String**, *Scode* **As Long**, *Source* **As String**, *HelpFile* **As String**, *HelpContext* **As Long**, *CancelDisplay* **As Boolean**)

Parameters

ErrCode

The short error code. For a list of possible NNTP error codes see the NNTP Error Code section of the Error Codes and Messages Appendix.

Description

String containing error information.

sCode

The long Scode.

Source

Error source.

HelpFile

Help file name.

HelpContext

Help file context.

CancelDisplay

Indicates whether to cancel the display. The default value for *CancelDisplay* is False meaning you do want to use the default message box. If you does not want to display the default error message box, set *CancelDisplay* to True.

RemoteHostName HTTP Server Event

Description

After a session has been accepted, the Internet protocol Server starts to asynchronously resolve the remote host IP address to its official name. When the resolution is complete, the RemoteHostName event is fired.

Syntax

*object*_**RemoteHostName** (*Session* **As Object**, *Success* **As Boolean**)

Parameters

Session

The object on which data transfer happens.

Data Type: Object

Default Value: N/A

Success

If *Success* is TRUE, then session's RemoteHostName property holds remote host's official name.

Data Type: Boolean

Default Value: N/A

Request HTTP Server Event

Description

This event is fired after a request from client has been received. Request value can be determined from session's RequestString property.

Syntax

*object*_**Request** (*Session* **As Object,** *EnableDefault* **As Boolean**)

Parameters

Session

The object on which data transfer happens.

Data Type: Object

Param: IN

Default Value: N/A

EnableDefault

EnableDefault enables/disables default handling of the request which is protocol dependent. The default value for *EnableDefault* is TRUE.

Data Type: Boolean

Param: IN

Default Value: TRUE

StateChanged HTTP Server Event

Description

This event is fired after the state of the transport state for the *Session* has changed. The state is given in Session's State and StateString properties.

Syntax

*object*_**StateChanged** (*Session* **As Object**)

Parameters

Session

The object on which data transfer happens.

Data Type: Object

Default Value: N/A

TimeOut HTTP Server Event

Description

This event is fired when the timer for the *Session* has expired (no incoming data from the client without the timeout period). See Session object's Timeout property for mode details.

Syntax

*object*_**TimeOut** (*Session* **As Object**)

Parameters

Session

The object on which data transfer happens.

Data Type: Object

Default Value: N/A

HTTP Server Sample Session

The following is all the code necessary to write a proxy HTTP server. It uses one HTTP server control (HTTPSRProxy) and one HTTP client control (HTTPCT1).

```
Private Sub btnProxyStart_Click()
HTTPSRProxy.MaxConnections = max
 HTTPSRProxy.ListenPort = port
 HTTPSRProxy.Start
End Sub

Private Sub btnProxyStart_Click()
HTTPSRProxy.Stop False
End Sub

Private Sub HTTPSRProxy_StateChanged(ByVal Session As Object,_
ByVal State As Integer)
  If State = prcConnected Then
    Session.Timeout = 1 ' in seconds
  End If
End Sub

Private Sub HTTPSRProxy_Timeout(ByVal Session As Object)
  Session.Close
End Sub

Private Sub HTTPSRProxy_Request(ByVal Session As Object,_
EnableDefault As Boolean)
    EnableDefault = False
    Session.ReplyString = ""
    If Session.Method = prsPut Or Session.Method = prsPost_
       Then ' POST and PUT
      httpct1.Method = Session.Method
      Session.DocInput.DocLink = httpct1.DocOutput.DocLink
      httpct1.DocInput.DocLink = Session.DocOutput.DocLink
      Session.ReplyDoc
      httpct1.SendDoc txtProxy & Session.Document
    Else ' GET and HEAD
      httpct1.Method = Session.Method
      Session.DocInput.DocLink = httpct1.DocOutput.DocLink
      Session.ReplyDoc
      httpct1.GetDoc txtProxy & Session.Document
    End If
End Sub

Private Sub HTTPSRProxy_DocInput(ByVal Session As Object,_
ByVal DocInput As DocInput)
  If DocInput.State = icDocHeaders Then
    If Len(Session.ReplyString) = 0 Then ' replystring hasn't been set
      Session.ReplyString = httpct1.ReplyString
    End If
  ElseIf DocInput.State = icDocData Then
    If Session.Method = prsHead Then ' for HEAD
      DocInput.SetData "" ' do not send data for HEAD request
    End If
  ElseIf DocInput.State = icDocEnd Then
    Session.DocInput.DocLink = Nothing ' for doclink
    httpct1.DocInput.DocLink = Nothing
    If Len(Session.ReplyString) = 0 Then ' didn't get_
       headers notification.
      Session.ReplyString = "HTTP/1.0 404 Error in processing"
      Session.ReplyDoc , "<BODY> Error in processing request,_
           please return to home page</BODY>"
    Else
      Session.Close
    End If
  End If
End Sub
```

Internet Client ActiveX Control Overview

The Internet Client ActiveX control implements the essential Internet client protocol entities and can be used to implement a new protocol at the application level.

Generally, a client application initiates connection to a server. After the connection has been established, the client may send or receive a small amount of data to or from the server. Either the client or the server can close the connection.

There are three types of transactions:

- **Requests** – A small amount of data that the client sends to the server. Requests are sent from the client to the server, who sends back a reply.
- **Replies** – A small amount of data that the server sends to the client
- **Data** – A large amount of data

Both a request and a reply are normally text based, whereas data can be either text or binary. In some cases, there is an application-specific string terminator for a reply. When incoming data contains the string terminator (for example "\r\n"), a data transaction is considered to be completed. Internet Client control is based on the this scenario. It has most of the common client properties, methods and events.

The following table describes the new properties, methods, and events that are specific to Internet Client control. For a complete list of properties, methods, and events, see Internet Client Properties, Method, and Events.

ReplyTerminator	The string terminator for replies. When the string is empty, closing the connection is considered to be the terminator.
DataTerminator	The string terminator for data. When the string is empty, closing the connection is considered to be the terminator.
ParsingMode	If ParsingMode is set to parsing reply (pmParsingReply), and the incoming data matches the ReplyTerminator, the Reply event is fired and the ReplyString property holds the full reply string from the server. If ParsingMode is set to parsing data (pmParsingData), all incoming data is streamed via the DocOutput event.
SendData	This method is used to send requests to the server.
PutDoc	This method is used to send large amount of data or a file to the server. Data is streamed via DocInput.
Reply event	The event is fired when parsing mode is set to parsing reply and there is a full reply available.

Internet Client Properties, Methods, and Events

The Internet Client ActiveX Control supports the following properties, methods and events. For an example illustrating the use of the control in a real life situation, see Internet Client Sample Session.

Property	Method	Event
Blocking	Cancel	Cancel
BlockResult	Connect	DocInput
DataTerminator	Disconnect	DocOutput
DocInput	PutDoc	Error

DocOutput	SendData	Log
EnableTimer		ProtocolStateChanged
Errors		Reply
LocalPort		StateChanged
Logging		TimeOut
NotificationMode		
ParsingMode		
ProtocolState		
ProtocolStateString		
RemoteHost		
RemotePort		
ReplyString		
ReplyTerminator		
SleepTime		
SocketHandle		
State		
StateString		
Timeout		

Blocking Internet Client Property

Description

Indicates whether methods should block until complete or not.

Syntax

object.**Blocking** *[=Boolean]*

Permission

W (Read/Write)

Availability

D (Design) and R (Runtime)

Data Type

Boolean.

Default Value

False.

Range

True or False.

Comments

If Blocking is set to True, then a method will not be completed until the proper event(s) for a given request are fired, if applicable.

BlockResult Internet Client Property

Description

Returns the result value of the last blocking method called.

Syntax

object.**BlockResult**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

BlockingResultConstants.

Default Value

icBlockOK..

Range

Name	Value	Description
icBlockOK	0	Blocking method was successful.
IcTimedOut	1	Blocking method returned due to timeout.
IcErrorExit	2	Blocking method returned due to an error.
IcBlockCancel	3	Blocking method returned due to cancel.
IcUserQuit	4	Blocking method returned due application end.

Comments

If Blocking is set to True, then a method will not be completed until the proper event(s) for a given request are fired, if applicable.

DataTerminator Internet Client Property

Description

The string terminator for data.

Syntax

object.**DataTerminator** *[= string]*

Permission

W (Read/Write)

Availability

D (Design) and R (Runtime).

Data Type

BSTR.

Default Value

“/r/n”.

Range

N/A

DocInput Internet Client Property

Description

Object describing input information for the document being transferred.

Syntax

object.**DocInput**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

DocInput.

Default Value

N/A.

Range

N/A.

Comments

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

Properties of the DocInput object may be set before calling the SendDoc method or they may be passed as arguments to this method. The DocInput object is also used for conveying information about the progress of the document transfer and for data linking and streaming.

For more information, see DocInput Object Overview and Common Control Objects.

DocOutput Internet Client Property

Description

Object describing output information for the document being transferred.

Syntax

object.**DocOutput**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

DocOutput.

Default Value

N/A.

Range

N/A.

Comments

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

Properties of the DocOutput object may be set before calling the GetDoc method or they may be passed as arguments to this method. The DocOutput object is also used for conveying information about the progress of the document transfer, and for data linking and streaming.

For more information, see DocOutput Object Events and Common Control Objects.

EnableTimer Internet Client Property

Description

Enable timer for the specified event. The event is specified by entering:

```
    EnableTimer(short event)
```

Syntax

object.**EnableTimer (***event***)** [= *Boolean*]

Permission

W (Write Only).

Note: This is the only control property that is Write only.

Availability

R (Runtime)

Data Type

Boolean.

Default Value

False. (The timer for this event will not be enabled.)

Range

True or False

Comments

Event is an integer value that determines the type of Timeout event that will be enabled. Constants defined for enum types for events are:

Value	Meaning
prcConnectTimeout = 1	Timeout for connect. If connection is not established within the timeout period, the Timeout event will be activated.
prcReceiveTimeout = 2	Timeout for receiving data. If no data arrives within the timeout period, the Timeout event will be activated.
prcUserTimeout= 65	Timeout for user defined event. Use prcUserTimeout + [Integer] range for custom timeout events.

Errors Internet Client Property

Description

A collection of errors that can be accessed for details about the last error that occurred. This collection should be used within an Error event if information passed through the Error event is not sufficient. For more details, see icErrors.

Syntax

object.**Errors**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

icErrors.

Default Value

N/A.

Range

N/A.

LocalPort Internet Client Property

Description

Designates the local port to use.

Syntax

object.**LocalPort** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

0.

Range

0 - 65535

Logging Internet Client Property

Description

Indicates whether log events should be fired when log data is available.

Syntax

object.**Logging** [= *Boolean*]

Permission

W (Read/Write).

Availability

D (Design) and R(Runtime).

Data Type

Boolean.

Default Value

False.

Range

True or False

NotificationMode Internet Client Property

Description

Determines when notification is issued for incoming data. Notification can also be suspended.

Syntax

object.**NotificationMode** [= *Integer*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

NotificationModeConstants.

Default Value

1.

Range

0-maximum unsigned long. At present, the values are:

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

ParsingMode Internet Client Property

Description

Describes the current parsing mode.

Syntax

object.**ParsingMode** *[=integer]*

Permission

W (Read/Write)

Availability

D (Design) and R (Runtime).

Data Type

ParsingModeConstants.

Default Value

PMParsingReply.

Range

Name	Value	Description
PMParsingReply	0	ParsingReply. Incoming data will be treated as replies.
PMParsingData	1	ParsingData. Incoming data will be treated as data.

ProtocolState Internet Client Property

Description

This property specifies the current state of the protocol.

Syntax

object.**ProtocolState**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Integer.

Default Value

"BASE".

Range

0-1. Constants defined for the enum types of ProtocolState property are:

Value	Meaning
BASE = 0	Base state before connection to server is established.
TRANSACTION =1	Connection to server is established. This is the valid state for calling methods on this control.

ProtocolStateString Internet Client Property

Description

String representation of ProtocolState.

Syntax

object.**ProtocolStateString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

BASE.

Range

N/A.

RemoteHost Internet Client Property

Description

The remote machine to connect to if the remoteHost parameter in the Connect method is missing. You can either provide a host name or an IP address string in dotted format. For example, “127.0.0.1.”

Note: This is the default property of the control.

Syntax

object.**RemoteHost** [= *String*]

Permission

W (Read/Write).

Availability

D (Design).

Data Type

String.

Default Value

Empty.

Range

N/A.

RemotePort Internet Client Property

Description

The remote port number to which to connect.

Syntax

object.**RemotePort** [= *Long*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

Long.

Default Value

8889

Range

1-65535.

ReplyString Internet Client Property

Description

Lists the last reply string sent by the server to the client as a result of a request.

Syntax

object.**ReplyString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

ReplyTerminator Internet Client Property

Description

The string terminator for parsing replies

Syntax

object.**ReplyTerminator** *[=string]*

Permission

W (Read/Write)

Availability

D (Design) and R (Runtime).

Data Type

BSTR.

Default Value

“\r\n”.

Range

N/A.

SleepTime Internet Client Property

Description

Specifies the sleep time between checking messages, if Blocking is True.

Syntax

object.**SleepTime** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

10 ms.

Range

>=zero.

Comments

Only applies when in Blocking mode.

SocketHandle Internet Client Property

Description

Socket handle for the primary connection (Request/Reply connection).

Syntax

object.**SocketHandle**

Permission

R (Read only)

Availability

R (Runtime).

Data Type

Long.

Default Value

N/A.

Range

>=0

Comments

Some protocols require more than one connection. SocketHandle is the handle for the request/reply connection.

State Internet Client Property

Description

This property specifies the connection state of the control.

Syntax

object.**State**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

StateConstants.

Default Value

prcDisconnected.

Range

1-6. Constants defined for enum types of State property are:

Value	Meaning
prcConnecting = 1	Connecting. Connect has been requested, waiting for connect acknowledge.
prcResolvingHost = 2	Resolving Host. Occurs only when RemoteHost is in name format rather than dot-delimited IP format.
prcHostResolved = 3	Resolved the host. Occurs only if ResolvingHost state has been entered previously.
prcConnected = 4	Connection established.
prcDisconnecting = 5	Connection closed. Disconnect has been initiated.
prcDisconnected = 6	Initial state when protocol object is instantiated, before Connect has been initiated, after a Connect attempt failed or after Disconnect performed.

StateString Internet Client Property

Description

A string representation of State.

Syntax

object.**StateString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

"Disconnected".

Range

N/A.

Timeout Internet Client Property

Description

Timeout value for the specified event. The event is specified by entering:

```
Timeout(short event)
```

Syntax

object.**Timeout** (*event*) [= *Long*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

Long.

Default Value

0.

Range

0-maximum unsigned long.

Comments

See Event values for EnableTimer .

Cancel Internet Client Method

Description

Cancels a pending request and disconnects the session.

Return Value

Void.

Syntax

object.**Cancel**

Parameters

None.

Connect Internet Client Method

Description

Initiates a Connect request. The control calls the StateChanged event if a connection is established.

Return Value

Void.

Syntax

object.**Connect** [*RemoteHost*], [*RemotePort*]

Parameters

RemoteHost

Optional. Remote host to which to connect. This arguments overrides the values from the corresponding RemoteHost property. The value of the RemoteHost property is not changed. If this parameter is missing, the control connects to the host defined in the RemoteHost property.

Data Type: BSTR

Param: IN

Default Value: N/A

RemotePort

Optional. Remote port to which to connect. This arguments overrides the values from the corresponding RemotePort property. The values of the RemotePort property is not changed. If this parameter is missing, the control connects to the port defined in the RemotePort property.

Data Type: Long

Param: IN

Default Value: N/A

Disconnect Internet Client Method

Description

Disconnects the connection.

Return Value

Void.

Syntax

object.**Disconnect**

Parameters

None.

PutDoc Internet Client Method

Description

Initiates sending of a document or a large amount of data.

Return Value

Void.

Syntax

object.**PutDoc** [*Headers*], [*InputData*], [*InputFile*]

Parameters

Header

Optional. Header used for sending the document. This argument only applies to protocols where document headers can be sent.

Data Type: Docheaders.

Param: IN

Default Value: DocInput. Headers

Inputdata

Optional. A data buffer containing the document to be sent

Data Type: VARIANT.

Param: IN

Default Value: DocInput. GetData

InputFile

Optional. A local file containing the file to be sent.

Data Type: BSTR.

Param: IN

Default Value: DocInput. FileName

Comments

The PutDoc method allows sending (posting or putting) a document. For basic use of this method, arguments should be passed to PutDoc 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 properties of the DocInput and DocOutput objects can be set before calling PutDoc. The DocInput and DocOutput events can also be used for transfering data using streaming rather than local files.

See DocInput Object Overview and DocOutput Object Overview for additional information.

SendData Internet Client Method

Description

Sends small amounts of data to client. Large amounts of data should be sent using DocStream. (See DocInput and PutDoc for more information.)

Return Value

Void.

Syntax

object.**SendData** *data*

Parameters

Data

Data to be sent

Data Type: VARIANT.

Param: IN

Default Value: N/A

Internet Client Events

Events indicate that an action has been requested and processed. Any errors which occur during command processing result in the Error event being called with appropriate error codes. Error codes, state changes, and protocol return values can be checked during event processing.

The following sections describe the events supported by the Internet Client Control. Each description includes the syntax, related parameters, their data type, default value, and whether the parameter is used for input or output (IN or OUT). For a complete listing of events, see Internet Client Properties, Methods, and Events.

Cancel Internet Client Event

Description

This event is activated after a cancellation request has been completed and satisfied. After this event the object's state changes to Base.

Syntax

*object*_**Cancel**

Parameters

None.

DocInput Internet Client Event

Description

A DocInput related event that indicates the input data has been transferred or the DocInput state has changed.

Syntax

*object*_**DocInput** (*DocInput* **As DocInput**)

Parameters

DocInput

Object describing document input data for the current transfer.

Data Type: DocInput

Param: IN

Default Value: N/A

Comments

The DocInput event can be used in its basic form for notification of transfer progress, (for example, for updating a progress bar). The DocInput.BytesTotal, DocInput.BytesTransferred and DocInput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

To gain more power from this control, you can also use the DocInput event for data streaming. For more information, see DocInput Object Overview.

DocOutput Internet Client Event

Description

A DocOutput related event indicating that output data has been transferred or the DocOutput state has changed.

Syntax

*object*_**DocOutput** (*DocOutput* **As DocOutput**)

Parameters

DocOutput

Object describing document output data for the current transfer.

Data Type: DocOutput

Param: IN

Default Value: N/A

Comments

The DocOutput event can be used in its basic form to notify the user of transfer progress, (for example, for updating a progress bar). The DocOutput.BytesTotal, DocOutput.BytesTransferred and DocOutput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

To gain more power from this control, you can also use the DocOutput event for data streaming. For more information, see DocInput Object Overview.

Error Internet Client Event

Description

This event is activated when an error occurs in background processing (for example, failed to connect or failed to send or receive in the background).

Syntax

*object*_**Error** (*ErrCode* **As Integer**, *Description* **As String**, *Scode* **As Long**, *Source* **As String**, *HelpFile* **As String**, *HelpContext* **As Long**, *CancelDisplay* **As Boolean**)

Parameters

ErrCode

The short error code. For a list of possible error codes, see Error Codes and Messages.

Description

String containing error information.

sCode

The long Scode.

Source

Error source.

HelpFile

Help file name.

HelpContext

Help file context.

CancelDisplay

Indicates whether to cancel the display. The default is TRUE (no display of the default error message box). If you do want to use the default message box, set CancelDisplay to FALSE.

Log Internet Client Event

Description

This event is fired when logging data is available.

Syntax

*object*_**Log**

Parameters

None.

ProtocolStateChanged Internet Client Event

Description

This event is activated whenever the protocol state changes.

Syntax

*object*_**ProtocolStateChanged** (*Protocol State* **As Integer**)

Parameters

Refer to the ProtocolState property and ProtocolStateString for possible values of the Protocol state parameter.

StateChanged Internet Client Event

Description

This event is fired after the state of the transport state has changed.

Syntax

*object*_**StateChanged** (*State* **As Integer**)

Parameters

Refer to State Internet Client Property for possible values of the State parameter.

TimeOut Internet Client Event

Description

This event is fired when the timer for the specific events has expired .

Syntax

*object*_**TimeOut** (*Event* **As Integer**)

Parameters

Event.

Refer to the EnableTimer event for possible values of the event parameter.

Continue:

Determines if the timer is active or not for the given event. Set Continue to TRUE to keep the timer active.

Data Type

Boolean.

Default Value

False.

Reply Internet Client Event

Description

This event is activated when a full reply is available. ReplyString property contains the full reply.

Syntax

*object*_**Reply**

Parameters

None.

Internet Client Sample Session

The following code shows how to use Internet Client control to implement a Finger protocol.

```
Private Sub btnFinger_Click()
 txtReply = ""
 Finger.Connect txtHost, 79 ‘ connect to finger port
End Sub

Private Sub Finger_StateChanged(ByVal State As Integer)
 If State = prcConnected Then
  Finger.ReplyTerminator = "" ‘ closing of the connection
                                    ‘ is terminator
  Finger.ParsingMode = pmParsingReply
  Finger.SendData txtUser.Text & vbCrLf     ‘ sends out username
 End If
End Sub

Private Sub Finger_Reply()
      txtReply = Finger.ReplyString ‘ gets reply back, display it.
End Sub
```

Internet Server ActiveX Control Overview

The Internet Server ActiveX control is based on generic protocol server specifications and implements all property, methods, and events using the semantics defined in the generic protocol server control.

The following table lists the properties, methods, and events supported by the Internet Server Control. For an example illustrating the use of the control in a real life situation, see Internet Server Sample Session.

Property	Method	Event
Server		
Blocking	CloseAll	Accept
BlockResult	AboutBox	CloseAll
Errors	Start	DocInput
LocalPort	Stop	DocOutput
ListenPort		Error
MaxConnections		RemoteHostName
		Request
SleepTime		StateChanged
SocketHandle		TimeOut
Sessions Collection		
Count	Item	
Session Item		
DataTerminator	Close	
DocInput	ReplyDoc	
DocOutput	SendData	
NotificationMode		
ParsingMode		
RemoteHostIP		
RemoteHostName		
RemotePort		
ReplyString		
RequestTerminator		
State		
StateString		
ThreadID		
Timeout		

Blocking Internet Server Property

Description

Indicates whether methods should block until complete or not.

Syntax

object.**Blocking** *[=Boolean]*

Permission

W (Read/Write)

Availability

D (Design) and R (Runtime)

Data Type

Boolean.

Default Value

False.

Range

True or False.

Comments

If Blocking is set to True, then a method will not be completed until the proper event(s) for a given request are fired, if applicable.

BlockResult Internet Server Property

Description

Returns the result value of the last blocking method called.

Syntax

object.**BlockResult**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

BlockingResultConstants.

Default Value

icBlockOK..

Range

Name	Value	Description
icBlockOK	0	Blocking method was successful.
IcTimedOut	1	Blocking method returned due to timeout.
IcErrorExit	2	Blocking method returned due to an error.
IcBlockCancel	3	Blocking method returned due to cancel.
IcUserQuit	4	Blocking method returned due application end.

Comments

If Blocking is set to True, then a method will not be completed until the proper event(s) for a given request are fired, if applicable.

Errors Internet Server Property

Description

A collection of errors that can be accessed for details about the last error that occurred. This collection should be used within an Error event if information passed through the Error event is not sufficient. For more details, see icErrors.

Syntax

object.**Errors**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

icErrors.

Default Value

N/A.

Range

N/A.

ListenPort Internet Server Property

Description

The port number to listen on.

Syntax

object.**ListenPort** [= Integer]

Permission

W (Read/Write).

Availability

D (Design).

Data Type

Long.

Default

0.

Range

0-65535.

LocalPort Internet Server Property

Description

Designates the local port to use.

Syntax

object.**LocalPort** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

0.

Range

0 - 65535

MaxConnections Internet Server Property

Description

The maximum number of clients allowed to connect to the server. If connecting clients exceeds MaxConnections, the Error event is fired and new connections are rejected until the maximum number of connections drops below the MaxConnection limit.

Syntax

object.**MaxConnections** [= *Integer*]

Permission

W (Read/Write).

Availability

D (Design).

Data Type

Long.

Default Value

100.

Range

0-65535

Sessions Internet Server Property

Description

The server session collection object. Session object provides access to a connection accepted from a client. Sessions can be enumerated and actions can be performed on them. Normally a session object is passed as an argument to an event and the action can be performed on the object.

Syntax

object.**Sessions** [= *Object*]

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Object.

Default Value

N/A.

Range

N/A.

SleepTime Internet Server Property

Description

Specifies the sleep time between checking messages, if Blocking is True.

Syntax

object.**SleepTime** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

10 ms.

Range

>=zero.

Comments

Only applies when in Blocking mode.

SocketHandle Internet Server Property

Description

Socket handle for the primary connection (Request/Reply connection).

Syntax

object.**SocketHandle**

Permission

R (Read only)

Availability

R (Runtime).

Data Type

Long.

Default Value

N/A.

Range

>=0

Comments

Some protocols require more than one connection. SocketHandle is the handle for the request/reply connection.

Count Sessions Collection Property (Internet Server)

Description

The number of items in the Sessions collection.

Syntax

object.**Count** [= *Integer*]

Permission

R (Read-only).

Availability

R(Runtime).

Data Type

Long.

Default Value

N/A.

Range

0-65535

DataTerminator Session Item Property

Description

Data terminator of the session

Syntax

object.**DataTerminator** [= *string]*

Permission

W (Read/Write).

Availability

R(Runtime).

Data Type

BSTR

Default Value

“\r\n”.

Range

N/A

DocInput Session Item Property (Internet Server)

Description

Object describing input information for the document being transferred.

Syntax

object.**DocInput**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

DocInput.

Default Value

N/A.

Range

N/A.

Comments

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

Properties of the DocInput object may be set before calling the SendDoc method or they may be passed as arguments to this method. The DocInput object is also used for conveying information about the progress of the document transfer and for data linking and streaming.

For more information, see DocInput Object Overview.

DocOutput Session Item Property (Internet Server)

Description

Object describing output information for the document being transferred.

Syntax

object.**DocOutput**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

DocOutput.

Default Value

N/A.

Range

N/A.

Comments

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

Properties of the DocOutput object may be set before calling the GetDoc method or they may be passed as arguments to this method. The DocOutput object is also used for conveying information about the progress of the document transfer, and for data linking and streaming.

For more information, see DocOutput DocOutput Object Event and Common Control Objects.

NotificationMode Session Item Property (Internet Server)

Description

Determines when notification is issued for incoming data. Notification can also be suspended.

Syntax

object.**NotificationMode** [= *NotificationMode Constant*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

NotificationConstants.

Default Value

icContinuousMode.

Range

Name	Value	Description
icCompleteMode	0	Notification is provided when there is a complete response.
icContinuousMode	1	Notification event is repeatedly activated when new data arrives.

ParsingMode Session Item Property

Description

Parsing mode of the session.

Syntax

object.**ParsingMode** [= *Integer*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

ParsingModeConstants.

Default Value

pmParsingRequest.

Range

Constants defined for enum types for ParsingMode property:

Name	Value	Description
pmParsingRequest	0	Parsing Request. The incoming data will be treated as requests.
pmParsingData	1	Parsing Data. The incoming data will be treated as data.

RemoteHostIP Session Item Property (Internet Server)

Description

Remote host IP address string for the connected session. Can be used by the server to display information about all sessions.

Syntax

object.**RemoteHostIP** [= *String*]

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

BSTR.

Default Value

N/A

Range

N/A.

RemoteHostName Session Item Property (Internet Server)

Description

Remote host's official name for the connected session. Can be used by the server to display information about all sessions. This property can be examined after the RemoteHostName event is successfully fired. See RemoteHostName event for additional information.

Syntax

object.**RemoteHostIP** [= *String*]

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

BSTR.

Default Value

N/A

Range

N/A.

RemotePort Session Item Property (Internet Server)

Description

The remote port number to which to connect. Can be used by the server to display information about all sessions.

Syntax

object.**RemotePort** [= *Long*]

Permission

R (Read-only).

Availability

R (Runtime)

Data Type

Long.

Default Value

N/A.

Range

0-65535.

ReplyString Session Item Property (Internet Server)

Description

Buffer contains an ASCII reply string for the session.

Syntax

object.**ReplyString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

RequestTerminator Session Item Property

Description

Request terminator of the session

Syntax

object.**RequestTerminator** [= *string]*

Permission

W (Read/Write).

Availability

R(Runtime).

Data Type

BSTR

Default Value

“\r\n”

Range

N/A

State Session Item Property (Internet Server)

Description

This property specifies the connection state of the control.

Syntax

object.**State**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Integer.

Default Value

prcDisconnected.

Range

1-6. Constants defined for enum types of State property are:

Value	Meaning
prcConnecting = 1	Connecting. Connect has been requested, waiting for connect acknowledge.
prcResolvingHost = 2	Resolving Host. Occurs when RemoteHost is in name format rather than dot-delimited IP format.
prcHostResolved = 3	Resolved the host. Occurs only if ResolvingHost state has been entered previously.
prcConnected = 4	Connection established.
prcDisconnecting = 5	Connection closed. Disconnect has been initiated.
prcDisconnected = 6	Initial state when protocol object is instantiated, before Connect has been initiated, after a Connect attempt failed or after Disconnect performed.

StateString Session Item Property (Internet Server)

Description

A string representation of State.

Syntax

object.**StateString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

"Disconnected".

Range

N/A.

ThreadID Session Item Property (Internet Server)

Description

Sessions thread identifier

Syntax

object.**ThreadID**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

N/A.

Range

N/A.

Timeout Session Item Property (Internet Server)

Description

The length of time that a connected session has no activity before a Timeout event is fired. Timeout event is fired when a connected session has no activity (no request from a client). Normally, the client is prevented from timing out when a data transfer is taking place. When the Timeout event is fired, user may choose to log the client out by calling the Close method on the session. If the user did not choose to log the client out, the Timeout event will be fired again when there is still no activity within the Timeout period.

Syntax

object.**Timeout**[= *Long*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

Long.

Default Value

300 seconds (5 minutes).

Range

>=0

AboutBox Internet Server Method

Description

Shows information about this control.

Return Value

Void.

Syntax

object.**AboutBox**

Parameters

None.

CloseAll Internet Server Method

Description

Close all connected sessions. The CloseAll event is fired.

Return Value

Void.

Syntax

object.**CloseAll**

Parameters

None

Start Internet Server Method

Description

Opens the server up for listening on the port specified by the ListenPort property.

Return Value

Void.

Syntax

object.**Start**

Parameters

None

Stop Internet Server Method

Description

Server stops listening for incoming connections.

Return Value

Void.

Syntax

object.**Stop** [*DisconnectSession*]

Parameters

DisconnectSession

Optional. If DisconnectSession is true, all connected sessions are disconnected. If DisconnectSessions is false, connected sessions will not be closed.

Data Type: Boolean

Param: IN

Default Value: True

Item Sessions Collection Method (Internet Server)

Description

Returns the session object from the collection. This method is the default method for a collection.

Return Value

Object.

Syntax

object.**Item** index

Parameters

index

Index is an integer number.

Data Type: Variant

Param: IN

Default Value: None

Close Session Item Method (Internet Server)

Description

Close the session by disconnecting. Error event is fired in case of failure. StateChanged event is fired after the connection is closed.

Return Value

Void.

Syntax

object.**Close**

Parameters

None

ReplyDoc Session Item Method (Internet Server)

Description

Initiates sending of a document

Return Value

Void.

Syntax

object.**ReplyDoc** [*Header*] [*InputData*] [*InputFiler*]

Parameters

Header

Optional. Header used for sending the document. This argument only applies for protocols where document headers can be sent.

Data Type: Docheaders.

Param: IN

Default Value: DocInput. Headers

Inputdata

Optional. A data buffer containing the document to be sent

Data Type: VARIANT.

Param: IN

Default Value: DocInput. GetData

InputFile

Optional. A local file containing the document to be sent.

Data Type: BSTR.

Param: IN

Default Value: DocInput. FileName

Comments

The ReplyDoc method allows sending (posting or putting) a document.

For basic use of this method, 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 properties of the DocInput and DocOutput objects can be set before calling ReplyDoc. The DocInput and DocOutput events can also be used for transfering data using streaming rather than local files. See the DocInput and DocOutput properties, the DocInput and DocOutput events, and the separate DocInput and DocOutput object documentation for more information.

SendData Session Item Method (Internet Server)

Description

Send small amount of data to client. Large amount of data should be sent using DocStream. (See DocInput and ReplyDoc for additional information.)

Return Value

Void.

Syntax

object.**SendData** *data*

Parameters

Data

Data to be sent

Data Type: VARIANT.

Param: IN

Default Value: N/A

Internet Server Events

Events indicate that an action has been requested and processed. Any errors that occur during command processing result in the Error event being called with appropriate error codes. Error codes, state changes, and protocol return values can be checked during event processing.

The following topics describe the events supported by the Internet Server Control. Each description includes the syntax, related parameters, their data type, default value, and whether the parameter is used for input or output (IN or OUT). For a complete list of events, see Internet Server ActiveX Control Overview.

Accept Internet Server Event

Description

This event is fired when there is an incoming connection request.

Syntax

*object*_**Accept** (*Session* **As Object,** *AcceptConnection* **As Boolean**)

Parameters

Session

Session's property RemoteHostIP and RemotePort can be examined and used to decide whether to accept the session or not.

AcceptConnection

To reject the connection, *AcceptConnection* needs to be set to FALSE. The default value for *AcceptConnection* is TRUE

CloseAll Internet Server Event

Description

This event is fired after all sessions have been closed.

Syntax

*object*_**CloseAll**

Parameters

None

DocInput Internet Server Event

Description

A DocInput related event that indicates the input data has been transferred or the DocInput state has changed.

Syntax

*object*_**DocInput** (*Session* **As Object**, *DocInput* **As DocInput**)

Parameters

Session

The session object on which transfer of input data happens.

Data Type: Object

Param: IN

Default Value: N/A

DocInput

Object describing document input data for the current transfer.

Data Type: DocInput

Param: IN

Default Value: N/A

Comments

The DocInput event can be used in its basic form for notification of transfer progress, (for example, for updating a progress bar). The DocInput.BytesTotal, DocInput.BytesTransferred and DocInput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

To gain more power from this control, you can also use the DocInput event for data streaming. For more information, see DocInput Object Overview.

DocOutput Internet Server Event

Description

A DocOutput related event indicating that output data has been transferred or the DocOutput state has changed.

Syntax

*object*_**DocOutput** (*Session* **As Object**, *DocOutput* **As DocOutput**)

Parameters

Session

The object on which output data transfer happens.

Data Type: Object

Param: IN

Default Value: N/A

DocOutput

Object describing document output data for the current transfer.

Data Type: DocOutput

Param: IN

Default Value: N/A

Comments

The DocOutput event can be used in its basic form to notify the user of transfer progress, (for example, for updating a progress bar). The DocOutput.BytesTotal, DocOutput.BytesTransferred and DocOutput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

To gain more power from this control, you can also use the DocOutput event for data streaming. For more information, see DocInput Object Overview.

Error Internet Server Event

Description

This event is activated when an error occurs in background processing (for example, failed to connect or failed to send or receive in the background).

Syntax

*object*_**Error** (*ErrCode* **As Integer**, *Description* **As String**, *Scode* **As Long**, *Source* **As String**, *HelpFile* **As String**, *HelpContext* **As Long**, *CancelDisplay* **As Boolean**)

Parameters

ErrCode

The short error code. For a list of possible error codes see Error Codes and Messages.

Description

String containing error information.

sCode

The long Scode.

Source

Error source.

HelpFile

Help file name.

HelpContext

Help file context.

CancelDisplay

Indicates whether to cancel the display. The default value for *CancelDisplay* is False meaning you do want to use the default message box. If you does not want to display the default error message box, set *CancelDisplay* to True.

RemoteHostName Internet Server Event

Description

After a session has been accepted, the Internet protocol Server starts to asynchronously resolve the remote host IP address to its official name. When the resolution is complete, the RemoteHostName event is fired.

Syntax

*object*_**RemoteHostName** (*Session* **As Object**, *Success* **As Boolean**)

Parameters

Session

The object on which data transfer happens.

Data Type: Object

Param: IN

Default Value: N/A

Success

If *Success* is TRUE, then session’s RemoteHostName property holds remote host’s official name.

Data Type: Boolean

Param: IN

Default Value: N/A

Request Internet Server Event

Description

This event is fired after a request from client has been received. Request value can be determined from session's RequestString property.

Syntax

*object*_**Request** (*Session* **As Object,** *EnableDefault* **As Boolean**)

Parameters

Session

The object on which data transfer happens.

Data Type: Object

Param: IN

Default Value: N/A

EnableDefault

EnableDefault enables/disables default handling of the request which is protocol dependent. The default value for *EnableDefault* is TRUE.

Data Type: Boolean

Param: IN

Default Value: TRUE

StateChanged Internet Server Event

Description

This event is fired after the state of the transport state for the *Session* has changed. The state is given in Session's State and StateString properties.

Syntax

*object*_**StateChanged** (*Session* **As Object**, State as short).

Parameters

Session

The object on which data transfer happens.

Data Type: Object

Param: IN

Default Value: N/A

State

The current transport state.

Data Type

Short.

Parameters

IN.

Default Value

N/A.

TimeOut Internet Server Event

Description

This event is fired when the timer for the *Session* has expired (no incoming data from the client within the timeout period). See Session object's Timeout Session Item property for mode details.

Syntax

*object*_**TimeOut** (*Session* **As Object**)

Parameters

Session

The object on which data transfer happens.

Data Type: Object

Param: IN

Default Value: N/A

Internet Server Sample Session

Click on the buttons below to view sample sessions showing a Finger Server and an FTP Server.

{button ,JI(`NIA.HLP',`IDH_Finger_Server_Sample')}   **Finger Server Sample**
{button ,JI(`NIA.HLP',`IDH_FTP_Server_Sample')} **FTP Server Sample**

Finger Server Sample

The following is the code for a FINGER server. It uses one Internet Server control (INETSR). Note that INETSR has been renamed to FINGERSR in this sample.

```
Private Sub btnStart_Click()
 FINGERSR.ListenPort = 79
 FINGERSR.Start
 If Err.Number <> 0 Then
  Log "Start finger server error: " & Err.Description
 Else
  Log "Finger server started"
 End If
End Sub

Private Sub FINGERSR_StateChanged(ByVal Session As Object,_
ByVal State As Integer)
 Log "StateChanged: Session.ThreadID = " & Session.ThreadID & "_
 : State = " & State

 If State = prcConnected Then
  'Set the parsing mode and request terminator
  Session.ParsingMode = pmParsingRequest
  Session.RequestTerminator = vbCrLf
 End If
End Sub

Private Sub FINGERSR_Request(ByVal Session As Object,_
EnableDefault As Boolean)
 Dim RequstString As String
 Dim username As String
 Dim pos As Integer
 RequestString = Session.RequestString
 Log "Request = " & RequestString


 pos = InStr(RequestString, "/W")
 'Extract the username
 username = Mid(RequestString, pos + Len("/W") + 1, _
   Len(RequestString) - 1 - Len("/W") - Len(Session.RequestTerminator))

 'To be done
 'For now, we just return some hard coded messages. We will return the    'appropriate
information.
 If username = "" Then
  If (pos > 0) Then
   Session.SendData "You are requesting all users'_
           information with /W option" & vbCrLf
  Else
   Session.SendData "You are requesting all users'_
           information" & vbCrLf
  End If
 Else
  If (pos > 0) Then
   Session.SendData "You are requesting " & username & "'s_
           information with /W option" & vbCrLf
  Else
   Session.SendData "You are requesting " & username & "'s_
           information" & vbCrLf
  End If
 End If
 Session.Close
End Sub
```

See Also

{button ,JI(`NIA.HLP',`IDH_FTP_Server_Sample')}   **FTP Server Sample**

FTP Server Sample

The following is a part of the code for a proxy FTP server. It uses one Internet Protocol Server control (INETSR), one TCP control (TCP), and one FTP client control (FTPCT). The TCP control is used to implement the FTP server data connection.

```
Private Sub btnStart_Click()
INETSR.MaxConnections = max
INETSR.ListenPort = txtPort
INETSR.Start
End Sub

Private Sub INETSR_StateChanged(ByVal Session As Object,|
ByVal State As Integer)

      If (State = prcConnected) Then
             Load TCP

              ' Send greetings
             Session.SendData "220 FTP OCX server ready" & vbCrLf

             Session.ParsingMode = pmParsingRequest
             Session.RequestTerminator = vbCrLf

      ElseIf (State = prcDisconnected) Then
             Unload TCP
      End If
End Sub
```

See Also

{button ,JI(`NIA.HLP',`IDH_Finger_Server_Sample')}   **Finger Server Sample**

MIME (with UUEncode) ActiveX Control Overview

The MIME (Multipurpose Internet Mail Extensions) ActiveX control allows users to set the necessary headers ("From", "To", etc.), add the desired attachments, and enter some body text. This object is useful for developing applications that send and receive mail messages having MIME and/or UUEncoded attachments. It can be used in conjunction with the SMTP, NNTP, and POP ActiveX controls.

The MIME ActiveX control enables you to compose multi-part messages without having to write the necessary code to format and encode these messages. References for mail message formats and MIME extensions can be found in RFC822 and RFC1521, respectively. For NNTP-specific header information, please refer to RFC1036.

The following table lists the properties, methods, and events supported by the MIME Control. For an example illustrating the use of the control in a real life situation, see Using DocLink, Using PushStream, and Example of MIME Multipart Message.

Note: The Compose method can be used to create an output file or stream the output string through a mail messaging client, such as the SMTPCT ActiveX. Attachments, in the form of referenced physical files, are on an accessible hard disk.

Property	Method	Event
Attachments	Compose	DocInput
Blocking	Decode	DocOutput
BlockResult	Encode	Error
Body	Load	
DocInput	UUDecode	
DocOutput	UUEncode	
FileName		
From		
Headers		
SleepTime		
Subject		
SubType		
To		
Type		
Attachments Collection		
Count	Append	
	Clear	
	Item	
	Remove	
Attachment Item		
AttachmentSize	Save	
Body		
Description		
EncodingFormat		
FileName		
MIMEType		
MIMETypeString		
SubType		
Type		

MIME Object Model Details

When Compose() method is called, the following headers will automatically be overwritten/added to the existing headers in creating the message. If the user modified these headers prior to Compose(), the values will be overwritten.

- Date
- Mime-Version
- Content-Type
- Message-Id

The following headers should always be filled out by the user. For details on optional headers, please refer to RFC822 documentation.

- From or Sender (actual address of submitter)
- To or Cc or Bcc or Newsgroups (only one is required to send a message, although any combination of the three are allowed. One or more addresses can be entered into these headers, each address being separated by a comma from another address.)

Each attachment object will be stored in its decoded form if the method Save is used. Prior to the Compose(), the contents of the attachment is not encoded in any form. For performance reasons, no memory is allocated for actual file contents until Compose(), so the file to be attached must not be moved/deleted until after Compose() is completed.

The methods Encode() and Decode() are lower-level utility methods and are not needed when the higher level methods Compose() or Load() are used. These lower-level methods are useful as pure encoding/decoding utilities for a given file.

For multiple address headers (To Cc, Bcc, Newsgroups), when Headers.Add() is used, the new address will be appended to and not overwrite the existing header (if any). To make a fresh start, the user can use Remove prior to Add (ex: Headers.Remove(‘To”), and then do Headers.Add(“To”, “joe@xyz.com”))

If any of the MIMETypeConstants other than icOtherMIMEType are used, then the MIMETypeString and EncodingFormat properties of the attachment are automatically filled in. The user must provide the MIMETypeString (e.g. the content-type string for the attachment) and EncodingFormat if icOtherMIMEType is used.

There is an option to have a UUEncoded document directly in the body of the text, or as an attachment. By default, when the user selects icUUEncodeMIMEType, the encoding will be added as an attachment. To have the document as part of the body, use the EncodingFormat of icUUEncodeInBody when doing an Append to the Attachments collection.

Attachments MIME Property

Description

Collection of attachments. You can add, remove, or modify any attachment in this collection.

Syntax

object.**Attachments**

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

Collection.

Default Value

N/A.

Range

N/A.

Blocking MIME Property

Description

Indicates whether methods should block until complete or not.

Syntax

object.**Blocking** *[=Boolean]*

Permission

W (Read/Write)

Availability

D (Design) and R (Runtime)

Data Type

Boolean.

Default Value

False.

Range

True or False.

Comments

If Blocking is set to True, then a method will not be completed until the proper event(s) for a given request are fired, if applicable.

BlockResult MIME Property

Description

Returns the result value of the last blocking method called.

Syntax

object.**BlockResult**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

BlockingResultConstants.

Default Value

icBlockOK..

Range

Name	Value	Description
icBlockOK	0	Blocking method was successful.
IcTimedOut	1	Blocking method returned due to timeout.
IcErrorExit	2	Blocking method returned due to an error.
IcBlockCancel	3	Blocking method returned due to cancel.
IcUserQuit	4	Blocking method returned due application end.

Comments

If Blocking is set to True, then a method will not be completed until the proper event(s) for a given request are fired, if applicable.

Body MIME Property

Description

The text part of the message that is neither a header nor an attachment. The body can be empty.

Syntax

object.**Body** [=*string*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

BSTR.

Default Value

Empty.

Range

N/A.

DocInput MIME Property

Description

Object describing input information for the document being transferred.

Syntax

object.**DocInput**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

DocInput

Default Value

N/A.

Range

N/A.

Comments

The DocInput object provides a more powerful interface beyond the basic capabilities of the SendDoc method. For basic use of the control, knowledge or use of the DocInput object is not required.

Properties of the DocInput object may be set before calling the SendDoc method of the control, or they may be passed as arguments to this method. The DocInput object is also used for conveying information about the progress of the document transfer, for data linking and data streaming. For more information, DocInput and DocOutput Objects.

DocOutput MIME Property

Description

Object describing output information for the document being transferred.

Syntax

object.**DocOutput**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

DocOutput.

Default Value

N/A.

Range

N/A.

Comments

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

Properties of the DocOutput object may be set before calling the GetDoc method or they may be passed as arguments to this method. The DocOutput object is also used for conveying information about the progress of the document transfer, and for data linking and streaming.

Filename MIME Property

Description

Name of the file parsed using Load(), or the output file (optional) for the Compose() method.

Syntax

object.**Filename** [=*string*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

BSTR.

Default Value

Empty.

Range

N/A.

From MIME Property

Description

Specifies who the mail message is from (from the mail message “From” header field). This value is actually stored in the DocOutput.Headers collection, but is exposed here as a convenience.

Syntax

object.**From** [=*string*]

Permission

W (Read/Write).

Availability

D (Design) and R(Runtime).

Data Type

BSTR.

Default Value

Emtpy.

Range

N/A.

Headers MIME Property

Description

A collection of headers that can be associated with a message. In order for a valid message to be comosed, the “From” or “Sender” header (one or both) must be set, as well as one or more of the following headers: “To,” “Cc,” “Bcc,” or “Newsgroups.” Newsgroups is used to send mail to newsgroups via the NNTP control..

Syntax

object.**Headers**

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Collection.

Default Value

Empty.

Range

N/A.

Comments

This Headers collection property maps directly with object DocOutput.Headers and could be used interchangeably. For more details about headers, see DocHeaders Collection Overview .

SleepTime MIME Property

Description

Specifies the sleep time between checking messages, if Blocking is True.

Syntax

object.**SleepTime** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

10 ms.

Range

>=zero.

Comments

Only applies when in Blocking mode.

Subject MIME Property

Description

Subject of the mail message (from the mail message “Subject” header field). This value is actually stored in the DocOutput.Headers collection, but is exposed here as a convenience.

Syntax

object.**Subject** [=*string*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

BSTR.

Default Value

N/A.

Range

N/A.

SubType MIME Property

Description

The message body's MIME SubType. Value is set after a Compose() or Load(). SubType is Read only.

Syntax

object.**SubType**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

BSTR.

Default Value

N/A.

Range

N/A.

To MIME Property

Description

Specifies who the mail message is to (from the mail message “To” header field). This value is actually stored in the DocOutput.Headers collection, but is exposed here as a convenience.

Syntax

object.**To =** [=*string*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

BSTR.

Default Value

Empty.

Range

N/A.

Type MIME Property

Description

The message body’s MIME type. Value is set after a Compose() or Load(). Type is read-only.

Syntax

object.**Type**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

BSTR.

Default Value

N/A.

Range

N/A.

Attachments Collection Property

The attachment object is a collection containing the attachments to a given message.

Count Attachments Collection Property

Description

The number of items in the collection.

Syntax

object.**Count**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

None.

Range

N/A.

Attachment Item Properties

The attachment object is an item in the attachments collection. Attachment items specify the filename, descriptive text, and MIME type of the given attachment.

AttachmentSize Attachment Item Property

Description

Size in bytes of the attachment.

Syntax

object.**AttachmentSize**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

None.

Range

N/A.

Comments

If the file is currently encoded, AttachmentSize is the encoded size.

Body Attachment Item Property

Description

Contents of the Attachment.

Syntax

object.**Body**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

BSTR.

Default Value

None.

Range

N/A.

Description Attachment Item Property

Description

Descriptive text specifying what the file contains. Optional, can be empty.

When attached by the Load() method, the description as given by the header Content-Description, if any (ex: Content-Description: 52-week chart). May be empty

Syntax

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

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

BSTR.

Default Value

None.

Range

N/A.

EncodingFormat Attachment Item Property

Description

Encoding format of the given fileContents of the Attachment.

Syntax

object.**EncodingFormat** [=*integer*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

EncodingFormatConstants as follows:

Constant	**Description**	**Encoding Format**
ic7BitEncode	NVT ASCII	7bit
icBase64Encode	Used mostly for binary files.	base64
icUUEncodeInBody	UUEncode, but place the result in body of message instead of as attachment.	UUEncode
icUUEncode	UUEncoded	UUEncode
icBinaryEncode	8-bit	base64
icOtherEncode	None of the above constants.	N/A

Default Value

None.

Range

N/A.

FileName Attachment Item Property

Description

The fully qualified pathname of the file to be attached when attaching, or, when attached by the Load() method, the name of the file, if given in the header “Content-Type”, sub-header “name” (ex: Content-Type: TEXT/plain; CHARSET=US-ASCII; name="final50.txt"). The Filename may be empty. If the Filename is empty, a temporary file name will be place here after the Load() method is called.

Syntax

object.**Filename** [=*string*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

BSTR.

Default Value

None.

Range

N/A.

MIMEType Attachment Item Property

Description

The content type/subtype of the file. This property is necessary to determine how to encode the file into a message. Not all registered content type/subtypes are supported by the MIME ActiveX Control. If this is not set by the user, the default is dependent on the extension of the file, if any. If there is no extension or if the extension is not typically associated with a given content type then the default would be icAppOctetStream.

Syntax

object.**MIMEType** [=*integer*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

MIMETypeConstants. See MIMETypeConstants.

Default Value

None.

Range

N/A.

MIMETypeString Attachment Item Property

Description

String value of MIMEType. If MIMEType is icOtherMimeType, this value is user-defined. Otherwise, it is one of the Description values shown in the MIMETypeConstants table.

Syntax

object.**MIMETypeString** [=*string*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

BSTR.

Default Value

None.

Range

N/A.

SubType Attachment Item Property

Description

String value of the Subtype portion of the MIMETypeString (i.e. “gif”, or “mpeg”.)

Syntax

object.**SubType**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

BSTR.

Default Value

None.

Range

N/A.

Type Attachment Item Property

Description

String value of Type portion of the MIMETypeString (such as "Application" or "Text").

Syntax

object.**Type**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

BSTR.

Default Value

None.

Range

N/A.

Headers Collection Property

The Headers object is a collection containing the headers for a MIME message. For more details about the Headers collection, please see DocHeaders Collection Overview and DocStream.

Count Headers Collection Property

Description

The number of items in the collection.

Syntax

object.**Count**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

None.

Range

N/A.

Header Item

The Header object is an item in the Headers collection. Header items specify the name and value of a given header.

Name Header Item Property

Description

The name of the header such as “Subject”, “From”, “To”, etc..

Syntax

object.**Name** [=*string*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

BSTR.

Default Value

None.

Range

N/A.

Value Header Item Property

Description

The value associated with the name, for example “Re: Upcoming activities”

Syntax

object.**Value** [=*string*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

BSTR.

Default Value

None.

Range

N/A.

Compose MIME Method

Description

Formats the information stored in the properties into a properly encoded mulitpart message. Returns 0 if successful and an Error Code (-1) if unable to properly create the message for any reason.

If a filename is included, the composed message will be stored in the file. However, if the filename is invalid (e.g. invalid directory or name), then the operation fails.

The composed message will always be streamed to the DocOutput event, the main message header appearing during the icDocHeaders state, and the rest of the data will appear during the icDocData state.

Return Value

0 or Error Code.

Syntax

object.**Compose** [*Filename*]

Parameters

Filename

Optional. The name of the file to be used to store the formatted message.

Data Type: String

Param: IN

Default Value: N/A

Decode MIME Method

Description

Given an encoded file, decode the information and write it to a file, if desired. Output will always be streamed to the DocOutput event. If the operation was unsuccessful, an error event will be fired and the return code will be non-zero.

Return Value

0 or Error Code.

Syntax

object.**Compose** M*imeType,[EncodingFormat], SourceFilename ,[DestinationFilename]*

Parameters

MimeType

Content type/subtype.

Data Type: MIMETypeConstants (see MIMETypeConstants.

Param: IN

Default Value: N/A

EncodingFormat

Optional. If MimeType is icOtherMimeType, then the EncodingFormat must not be empty.

Data Type: EncodingFormatConstants (see EncodingFormatConstants).

Param: IN

Default Value: N/A

SourceFilename

The fully qualified filename to be decoded.

Data Type: String

Param: IN

Default Value: None

DestinationFilename

Optional. The fully qualified filename to contain the decoded file.

Data Type: String

Param: IN

Default Value: N/A

Encode MIME Method

Description

Given a file to be encoded, encode the given information and write to a file, if desired. Output will always be streamed to the DocOutput event. If the operation was unsuccessful, an error event will be fired and the return code will be non-zero.

Return Value

0 or Error Code.

Syntax

object.**Encode** M*imeType,[EncodingFormat], SourceFilename [,DestinationFilename]*

Parameters

MimeType

Content type/subtype.

Data Type: MIMETypeConstants (see MIMETypeConstants).

Param: IN

Default Value: N/A

EncodingFormat

Optional. If MimeType is icOtherMimeType, then the EncodingFormat must not be empty.

Data Type: EncodingFormatConstants (see EncodingFormatConstants).

Param: IN

Default Value: N/A

SourceFilename

The fully qualified filename to be encoded.

Data Type: String

Param: IN

Default Value: None

DestinationFilename

Optional. The fully qualified filename to contain the encoded file.

Data Type: String

Param: IN

Default Value: N/A

Load MIME Method

Description

Given a multipart message either in the form of a BSTR or as a physical file name, populate the Headers and Attachments collections and Body property with the new information. The actual attachments themselves will not saved into a file unless the user explicitly uses the Save() method of the Attachment object. Returns 0 if successful or an Error Code (-1) if unable to properly decode the message for any reason.

Return Value

0 or Error Code.

Syntax

object.**Load** *Filename*

Parameters

Filename

The name of the file that contains the formatted message.

Data Type: String

Param: IN

Default Value: N/A

UUDecode MIME Method

Description

Given a Base64-encoded string, returns the decoded value.

Return Value

BSTR.

Syntax

object.**UUDecode** *InputString*

Parameters

InputString

The string to be decoded.

Data Type: String

Param: IN

Default Value: N/A.

Comments

This method is useful for user authentications, which require encoding and/or decoding Base 64 passwords and user names.

UUEncode MIME Method

Description

Given any Input string, return a Base64-encoded output string.

Return Value

BSTR.

Syntax

object.**UUEncode** *InputString*

Parameters

InputString

The string to be encoded.

Data Type: String

Param: IN

Default Value: N/A.

Comments

This method is useful for user authentications, which require encoding and/or decoding Base64 passwords and user names.

Append Attachment Collection Method

Description

Adds an attachment item to the collection. If an Attachment item with the same FileName currently exists, then the Description and MimeType will be updated with the new values.

Return Value

None.

Syntax

object.**Append** FileName, Description, M*imeType,[FileFormat], [MimeTypeString], [AttachmentSize]*

Parameters

FileName

Name of the file to be attached. When the message is actually composed in Compose(), the FileName will be stripped of its path components and added as a subheader to the Header “Content-Type”. The name of the sub-header is “name”.

Data Type: BSTR

Param: IN

Default Value: None

Description

Description of what the file contains.

Data Type: BSTR

Param: IN

Default Value: None

MimeType

Content type/subtype.

Data Type: MIMETypeConstants (see MIMETypeConstants).

Param: IN

Default Value: N/A

FileFormat

Optional. If MimeType is icOtherMimeType, then the FileFormat must not be empty.

Data Type: EncodingFormatConstants (see EncodingFormatConstants).

Param: IN

Default Value: N/A

MimeTypeString

Optional. If MimeType is icOtherMimeType, then the MimeTypeString must not be empty. MimeTypeString will be used to form the Content-Type body header. It should follow the type/sub-type format. Example: “application/x-javascript”.

Data Type: EncodingFormatConstants (see EncodingFormatConstants).

Param: IN

Default Value: N/A

AttachmentSize

Size of the file. If the text is encoded, the size of the file is the encoded size.

Data Type: Long

Param: IN

Default Value: None

Clear Attachment Collection Method

Description

Removes all attachments from the collection.

Return Value

None.

Syntax

object.**Clear**

Parameters

None

Item Attachment Collection Method

Description

Returns an item from the collection. The Item method is the default method for a collection.

Return Value

Index.

Syntax

object.**Item** *Index*

Parameters

Index

Index may be either an integer or a string. Integer indices identify an item by its 1-based index. String indices identify an item by its FileName property.

Data Type: VARIANT

Param: IN

Default Value: None

Remove Attachment Collection Method

Description

Removes an attachment from the collection

Return Value

None.

Syntax

object.**Remove** *Filename*

Parameters

Filename

The name of the file to be removed.

Data Type: BSTR

Param: IN

Default Value: N/A

Save Attachment Item Method

Description

Stores the current attachment into a file. If the attachment was encoded, then it will be decoded.

Return Value

0 or Error Code.

Syntax

object.**Save** *[Filename]*

Parameters

Filename

Name of file to write to. Can be path qualified. If none is used, then the value of the property FileName will be used. If it is empty, an error will be thrown.

Data Type: BSTR

Param: IN

Default Value: None

Add Headers Collection Method

Description

Adds a header to the collection. If the Name currently exists, then the Value will be updated with the new value (overwriten, not appended).

Return Value

None.

Syntax

object.**Add** *Name, Value*

Parameters

Name

Name of the header to be added.

Data Type: BSTR

Param: IN

Default Value: None

Value

Value of the header to be added.

Data Type: BSTR

Param: IN

Default Value: None

Clear Headers Collection Method

Description

Removes all the headers from the collection.

Return Value

None.

Syntax

object.**Clear**

Parameters

None

Item Headers Collection Method

Description

Returns an item from the collection. The Item method is the default method for a collection.

Return Value

Header.

Syntax

object.**Item** *Index*

Parameters

Index

Index may be either an integer or a string. Integer indices identify an item by its 1-based index. String indices identify an item by its Name property.

Data Type: VARIANT

Param: IN

Default Value: None

Remove Headers Collection Method

Description

Removes a given header from the collection.

Return Value

None.

Syntax

object.**Remove** *Index*

Parameters

Index

Index may be either an integer or a string. Integer indices identify an item by its 1-based index. String indices identify an item by its Name property.

Data Type: VARIANT

Param: IN

Default Value: None

Text Headers Collection Method

Description

Returns all the headers in text format.

Return Value

BSTR.

Syntax

object.**Text**.

Parameters

None.

MIME Events

Events are used for MIME notification. They indicate that an action has been requested and processed. Any errors which occur during command processing result in the Error event being called with appropriate error codes. Error codes, state changes, and protocol return values are usually checked during event processing.

The next series of Help topics describe the events supported by the MIME ActiveX Control. Each description includes the syntax, related parameters, their data type, default value, and whether the parameter is used for input or output (IN or OUT). For a complete listing of events, see MIME (with UUEncode) ActiveX Control Overview.

DocInput MIME Event

Description

A DocInput related event that indicates the input data has been transferred or the DocInput state has changed.

Syntax

*object*_**DocInput** (*DocInput* **As DocInput**)

Parameters

DocInput

Object describing document input data for the current transfer.

Data Type: DocInput

Param: IN

Default Value: N/A

Comments

The DocInput event can be used in its basic form for notification of transfer progress, (for example, for updating a progress bar). The DocInput.BytesTotal, DocInput.BytesTransferred and DocInput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

To gain more power from this control, you can also use the DocInput event for data streaming. For more information, see DocInput Object Overview and Common Control Objects.

DocOutput MIME Event

Description

A DocOutput related event indicating that output data has been transferred or the DocOutput state has changed.

Syntax

*object*_**DocOutput** (*DocOutput* **As DocOutput**)

Parameters

DocOutput

Object describing document output data for the current transfer.

Data Type: DocOutput

Param: IN

Default Value: N/A

Comments

The DocOutput event can be used in its basic form to notify the user of transfer progress, (for example, for updating a progress bar). The DocOutput.BytesTotal, DocOutput.BytesTransferred and DocOutput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

To gain more power from this control, you can also use the DocOutput event for data streaming. For more information, see DocOutput Object Overview.

Error MIME Event

Description

This event is activated when an error occurs in background processing (for example, failed to connect or failed to send or receive in the background).

Syntax

*object*_**Error** (*ErrCode* **As Integer**, *Description* **As String**, *Scode* **As Long**, *Source* **As String**, *HelpFile* **As String**, *HelpContext* **As Long**, *CancelDisplay* **As Boolean**)

Parameters

ErrCode

The short error code. For a list of possible MIME error codes see MIME Error Codes.

Description

String containing error information.

sCode

The long Scode.

Source

Error source.

HelpFile

Help file name.

HelpContext

Help file context.

CancelDisplay

Indicates whether to cancel the display. The default is TRUE (no display of the default error message box). If you do want to use the default message box, set CancelDisplay to FALSE.

MIMETypeConstants

The following table defines MIMETypeConstants.

Constant	Description (MimeTypeString)	Suffix	Encoding Format
icTextPlain	text/plain	.txt	7bit
icMultipartMixed	multipart/mixed		7bit
icAppTar	application/x-tar	.tar	base64
icAppPostscript	application/postscript	.ps	base64
icAppMsWord	application/msword	.doc	base64
icAppOctetStream	application/octet-stream		base64
icAppZip	application/zip	.zip	base64
icAppPdf	application/pdf	.pdf	base64
icImageJpeg	image/jpeg	.jpe,.jpg, .jpeg	base64
icImageBmp	image/ms-bmp	.bmp	base64
icImageGif	image/gif	.gif	base64
icImageTiff	image/tiff	.tif, .tiff	base64
icImageXBmp	image/x-bmp	.xbm	base64
icVideoMpeg	video/mpeg	.mpe,.mpeg,.mpg	base64
icVideoAvi	video/x-msvideo	.avi	base64
icVideoQuickTime	video/quicktime	.mov,.qt	base64
icAudioAu	audio/basic	.au,.snd	base64
icAudioWav	audio/x-wav	.wav	base64
icTextHtml	text/html	.html, .htm,.sht, .s html	7bit
icVrml	x-world/x-vrml	.wrl	7bit
icUUEncodeMIME Type	UUEncoded		UUEncode
icOtherMimeType	new mime-type, or one not in the list of constants. The MimeTypeString will be user-inputted.		user ‘s choice

EncodingFormatConstants

The following table defines Encoding Format Constants.

Constant	Description	Encoding Format
ic7BitEncode	NVT ASCII	7bit
icBase64Encode	Used mostly for binary files.	base64
icUUEncodeInBody	UUEncode, but place the result in body of message instead of as attachment.	UUEncode
icUUEncode	UUEncoded	UUEncode
icBinaryEncode	8-bit	base64
icOtherEncode	None of the above formats.	N/A.

MIME Sample Sessions

Click on one of the following examples to see sample sessions.

{button ,JI(`NIA.HLP',`IDH_Using_DocLink')} **Composing a message using DocLink** {button ,JI(`NIA.HLP',`IDH_Using_PushStream')} **Composing a message using PushStream**

{button ,JI(`NIA.HLP',`IDH_Receiving_a_Message')} **Receiving a message**

{button ,JI(`NIA.HLP',`IDH_Low_Level_Utilities')} **Low-level utilities using Encode and Decode**

{button ,JI(`NIA.HLP',`IDH_Example_of_MIME_Multipart_Message')} **Example of MIME multipart message**

Using DocLink

A code fragment for a simple application using the MIME ActiveX control to compose a message might look like this. It uses the SMTP ActiveX Control and NNTP ActiveX Control to send the mail. In this example, DocLink is used for streaming.

```
mime1.Headers.Add( “From”, “joe@xyz.com”)
mime1.Headers.Add(“To”, “CppUsers”)
mime1.Headers.Add(“Subject”, “C++ vs. Java”)
mime1.Headers.Add(“Newsgroups”, “alt.computers.lang”)

mime1.Body = “Here is a list of interesting articles for your perusal.”

mime1.Attachments.Append “c:\doc\article1.doc”, “Multi-Inheritance issues”, icAppMsWord
mime1.Attachments.Append “c:\doc\article2.txt”, “Memory Management”, icTextPlain
mime1.Attachments.Append “c:\doc\article3.jpg”, “humorous Dilbert cartoon”, icImageJpg
mime1.Attachments.Append “c:\doc\codesamp.zip”, “different code samples”, icAppZip

smtpct1.RemoteHost = “mail”
nntpct1.RemoteHost = “news”

smtpct1.DocInput.DocLink = mime1.DocOutput.DocLink ‘stream data to smtpct’s DocInput
nntpct1.DocInput.DocLink= mime1.DocOutput.DocLink  ‘stream data to nntpct’s DocInput
rc = mime1.Compose “\temp\result”
```

Using PushStream

This is the same example as that used in Using DocLink, except using PushStream instead of DocLink for streaming.

```
mime1.Headers.Add( “From”, “joe@xyz.com”)
mime1.Headers.Add(“To”, “CppUsers”)
mime1.Headers.Add(“Subject”, “C++ vs. Java”)
mime1.Headers.Add(“Newsgroups”, “alt.computers.lang”)

mime1.Body = “Here is a list of interesting articles for your perusal.”

mime1.Attachments.Append “c:\doc\article1.doc”, “Multi-Inheritance _ issues”, icAppMsWord
mime1.Attachments.Append “c:\doc\article2.txt”, “Memory Management”,_ icTextPlain
mime1.Attachments.Append “c:\doc\article3.jpg”, “humorous Dilbert _ cartoon”, icImageJpg
mime1.Attachments.Append “c:\doc\codesamp.zip”, “different code _
samples”, icAppZip

smtpct1.RemoteHost = “mail”
nntpct1.RemoteHost = “news”

InSendMethod = True
smtpct1.DocInput.PushStreamMode = True
nntpct1.DocInput.PushStreamMode = True

rc = mime1.Compose “\temp\result”    ‘ the file name is optional, use only
‘if you wish to save the composed file

‘ Now here is what happens in the MIME Docoutput event:

Private Sub MIME1_DocOutput(ByVal DocOutput As DocOutput)
 If (InSendMethod = True) Then
   If (DocOutput.state = icDocBegin) Then
    smtpct1.DocInput.PushStream
      nntpct1.DocInput.PushStream
   ElseIf (DocOutput.state = icDocHeaders) Then 'headers
     smtpct1.DocInput.Headers.Clear
      nntpct1.DocInput.Headers.Clear
     Dim header As DocHeader
     For Each header In DocOutput.Headers
       smtpct1.DocInput.Headers.Add header.Name, header.Value
         nntpct1.DocInput.Headers.Add header.Name, header.Value
     Next
     smtpct1.DocInput.PushStream
       nntpct1.DocInput.PushStream
   ElseIf (DocOutput.state = icDocData) Then 'DATA
     Dim buf As Variant
     DocOutput.GetData buf
     smtpct1.DocInput.SetData buf  ' send icdocdata
      nntpct1.DocInput.SetData buf  ' send icdocdata
      smtpct1.DocInput.PushStream
     nntpct1.DocInput.PushStream
   ElseIf (DocOutput.state = icDocEnd) Then
     smtpct1.DocInput.PushStream
      nntpct1.DocInput.PushStream
     InSendMethod = False
   ElseIf (DocOutput.state = icDocError) Then
     InSendMethod = False
  End If
End If

End Sub
```

Receiving a Message

A code fragment for a simple application using the MIME ActiveX to parse a message might look like this. It uses the POP ActiveX Control (DocOutput event) to receive the mail.

```
' Syntax:  You may use the second syntax if RemoteHost, UserId,
'Password set in properties correctly
  ' 1.)   //pop://user:password@hostname:portnum/message_number
  ' 2.)   //pop:///message_number
  ' 3.)   //pop://user:password@hostname:/message_number

  If (popct1.State = prcConnected) Then

      <some error logic code>

      'if already connected and user changes host, userid, or password
     'then goto AlreadyConnected. If this message box wasn't there and
     'Syntax formats 1.) or 3.) was used, then would get an error
     'event, 1013:
     ' "1013 : The argument passed to a function was not in the
     'correct format or in specified range"

     tmpstring = "pop:///" & txtMsgNumber.Text
     popct1.URL = tmpstring
  Else
  ' You can use rhost.Text,UserId.text, Password.Text directly
  ' the port number, 110, is optional
    On Error GoTo exit_sub
    popct1.URL = "pop://" + UserId.Text + ":" + Password.Text +
           "@" + rhost.Text + ":110/" + txtMsgNumber.Text
  End If

  Popct1.GetDoc , , “\temp\a_msg”   ‘write to a_msg

  AlreadyConnected:
       MsgBox "Cannot change URL properties when already connected._
       GetDoc command will not be executed."

‘ NOTE: once the GetDoc is completed, then the output file should be
‘      created. Wait for the icDocEnd event to occur inside of
‘      popct1_DocOutput, or if you were not previously connected to the
‘      mail server, you can wait until popct1_State event is
‘      prcDisconnected before doing Load.

Private Sub Popct1_DocOutput(ByVal DocOutput As DocOutput)
      If (DocOutput.State = icDocEnd) Then ‘finished
             mime1.Load “\temp\a_msg”
      End If
End Sub
```

Low-Level Utilities

The following are examples of using the MIME ActiveX as a simple encoding/decoding utility. The developer can use other tools to format or parse a mail message, if desired.

```
mime1.Encode icAppOctetStream, , “\temp\applet.exe”, “\temp\applet.enc”
mime1.Decode icAppOctetStream, , “\temp\applet.enc”, “\temp\applet.dec”
Dim EncodedResult as string
Dim DecodedResult as string
EncodedResult = MIME1.UUEncode “password”
DecodeResult = MIME1.UUDecode EncodedResult
```

Example of MIME Multipart Message

```
Date: Tue, 9 Apr 96 13:04:24
From: Joe User <Joeu>
Subject: files you need....
To: group-dist@xyz.com
Return-Receipt-To: joeu
X-MAILER: Chameleon 4.6, TCP/IP for Windows, NetManage Inc.
X-PRIORITY: 3 (Normal)
RETURN-RECEIPT-TO: joeu
Message-ID: <Chameleon.829083867.joeu@joeu>
MIME-Version: 1.0
Content-Type: MULTIPART/MIXED; BOUNDARY="joeu:829083867:828:157:41"

--joeu:829083867:828:157:41
Content-Type: TEXT/PLAIN; charset=US-ASCII

The attached files are sed.exe, grep.exe and idheader.bat.

thanks,
 Joe

--joeu:829083867:828:157:41
Content-Type: APPLICATION/OCTET-STREAM; SIZEONDISK=32034; NAME="sed.exe"
Content-Transfer-Encoding: BASE64
Content-Description: sed.exe

TVqDABcARAAjAPAK///AAYAAOeBjAMgBQAAAAAEAAAAAAAAAAAAAAAA
// this is the encoded file, sed.exe............
bnVsbCBwb2ludGVyIGFzc2lnbm1lbnQNCgD///8BAAIAUAYCAAAA

--joeu:829083867:828:157:41
Content-Type: APPLICATION/OCTET-STREAM; SIZEONDISK=17322; NAME="grep.exe"
Content-Transfer-Encoding: BASE64
Content-Description: grep.exe

TVqqASIAAAAAAAAAAAAAAgAAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFt
// this is the encoded file, grep.exe............
IGZsb2F0aW5nIHBvaW50IG5vdCBsb2FkZWQNCgD///8BAAIAlAMCAAAA

--joeu:829083867:828:157:41
Content-Type: TEXT/PLAIN; SIZEONDISK=5737; NAME="idr.bat"; CHARSET=US-ASCII
Content-Description: idr.bat

<here is text for the file idr.bat>

--joeu:829083867:828:157:41--  // terminator for the msg is boundary+“--”
```

NNTP Client ActiveX Control Overview

The NNTP (Networking News Transfer Protocol) Client Control provides a reusable component that allows applications to access NNTP news servers. It provides news reading and posting capabilities. NNTP implements the basic client NNTP Protocol as specified by RFC977, *Network News Transfer Protocol*, and implements NNTP extension commands as documented in the Internet-Draft on Common NNTP Extensions. For questions on Internet-Drafts contact, contact Internet-Drafts@CNRI.Reston.VA.US.

This control can be used by Visual Basic, Delphi, and C++ programmers to develop applications that communicate with NNTP servers to view news groups, retrieve and post news articles. For example, you can create an application that: browses selected newsgroups and retrieves articles.

The following table lists the properties, methods, and events supported by the NNTP Control in alphabetical order. NNTP uses a special type of authentication, which is reflected in the methods and events supported. For an example illustrating the use of the control in a real life situation, see NNTP Sample Session.

Property	**Method**	**Event**
ArticleNumbersSupported	AboutBox	ArticleStatus
Blocking	Cancel	AuthenticateRequest
BlockResult	Connect	AuthenticateResponse
Busy	GetAdministrationFile	Banner
	GetArticleByArticleNumber	Busy
DocOutput	GetArticleByMessageID	Cancel
EnableTimer	GetArticleHeaders	DocInput
Errors	GetArticleNumbers	DocOutput
LocalPort	GetBodyByArticleNumber	Error
LastUpdate NNTP Property	GetBodyByMessageID	LastArticle
Logging	GetDoc	Log
NotificationMode	GetHeaderByArticleNumber	NextArticle
OverviewSupported	GetHeaderByMessageID	ProtocolStateChanged
PostingAllowed	GetOverview	SelectGroup
ProtocolState	GetOverviewFormat	StateChanged
ProtocolStateString	GetStatByArticleNumber	TimeOut
RemoteHost	ListGroups	
RemotePort	ListGroupDescriptions	
ReplyCode	ListNewGroups	
ReplyString	Quit	
SleepTime	SelectGroup	
SocketHandle	SendDoc	
State	SetLastArticle	
StateString	SetNextArticle	
Timeout		
URL		

NNTP Commands

The following table summarizes the NNTP Client commands as specified in RFC977 and the NNTP extension commands as specified in Internet-Draft on Common NNTP Extensions.

NNTP Client Commands	**NNTP Extension Commands**
ARTICLE	AUTHINFO
GROUP	LISTGROUP

LIST	LIST OVERVIEW.FMT
NEWSGROUP	XHDR
POST	XMOTD
QUIT	XOVER

Using the NNTP Control

To use the NNTP Client ActiveX Control you must choose the NNTP toolbox icon.

There should be no speed overhead and response delay other than the one given by the network. This control uses and is dependent on DocStreams.

ArticleNumbersSupported NNTP Property

Description

If True, the GetArticleNumbers method may be used to retrieve a list of article numbers for a newsgroup. This property has no meaning before the connection to the server has been established.

Syntax

object.**ArticleNumbersSupported**

Permission

R (Read-only).

Availability

R(Runtime).

Data Type

Boolean.

Default Value

False.

Range

True or False

Comments

Examine this property after a connection is established to determine if the server supports the GetArticleNumber method (LISTGROUP command).

Blocking NNTP Property

Description

Indicates whether methods should block until complete or not.

Syntax

object.**Blocking** *[=Boolean]*

Permission

W (Read/Write)

Availability

D (Design) and R (Runtime)

Data Type

Boolean.

Default Value

False.

Range

True or False.

Comments

If Blocking is set to True, then a method will not be completed until the proper event(s) for a given request are fired, if applicable.

BlockResult NNTP Property

Description

Returns the result value of the last blocking method called.

Syntax

object.**BlockResult**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

BlockingResultConstants.

Default Value

icBlockOK.

Range

Name	**Value**	**Description**
icBlockOK	0	Blocking method was successful.
icTimedOut	1	Blocking method returned due to timeout.
icErrorExit	2	Blocking method returned due to an error.
icBlockCancel	3	Blocking method returned due to cancel.
icUserQuit	4	Blocking method returned due application end.

Busy NNTP Property

Description

Indicates a command is in progress.

Syntax

object.**Busy** *[= Boolean]*

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Boolean.

Default Value

N/A.

Range

True or False

Comments

The vlaue is True if the command is in progress and False if not.

DocInput NNTP Property

Description

Object describing input information for the document being transferred.

Syntax

object.**DocInput**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

DocInput.

Default Value

N/A.

Range

N/A.

Comments

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

Properties of the DocInput object may be set before calling the SendDoc method or they may be passed as arguments to this method. The DocInput object is also used for conveying information about the progress of the document transfer and for data linking and streaming.

For more information, see DocInput Object Overview and the Common Control Objects.

DocOutput NNTP Property

Description

Object describing output information for the document being transferred.

Syntax

object.**DocOutput**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

DocOutput.

Default Value

N/A.

Range

N/A.

Comments

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

Properties of the DocOutput object may be set before calling the GetDoc method or they may be passed as arguments to this method. The DocOutput object is also used for conveying information about the progress of the document transfer, and for data linking and streaming.

For more information, see DocOutput Object Overview DocOutput and Common Control Objects.

EnableTimer NNTP Property

Description

Enable timer for the specified event. The event is specified by entering:

```
EnableTimer(short event)
```

Syntax

object.**EnableTimer** *(event)* [= *Boolean*]

Permission

W (Write Only).

Note: This is the only control property that is Write only.

Availability

R (Runtime)

Data Type

Boolean.

Default Value

False. (The timer for this event will not be enabled.)

Range

True or False

Comments

Event is an integer value that determines the type of Timeout event that will be enabled. Constants defined for enum types for events are:

Value	Meaning
prcConnectTimeout = 1	Timeout for connect. If connection is not established within the timeout period, the Timeout event will be activated.
prcReceiveTimeout = 2	Timeout for receiving data. If no data arrives within the timeout period, the Timeout event will be activated.
prcUserTimeout= 65	Timeout for user defined event. Use prcUserTimeout + [Integer] range for custom timeout events.

Errors NNTP Property

Description

A collection of errors that can be accessed for details about the last error that occurred. This collection should be used within an Error event if information passed through the Error event is not sufficient. For more details, see icErrors.

Syntax

object.**Errors**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

icErrors.

Default Value

N/A.

Range

N/A.

LocalPort NTTP Property

Description

Designates the local port to use.

Syntax

object.**LocalPort** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

0.

Range

0 - 65535

LastUpdate NNTP Property

Description

The default value used by the GetAdministrationFile and ListNewGroups methods.

Syntax

object.**LastUpdate** [= *String*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

DATE.

Default

The time the control is first created by the ActiveX container, or ActiveX-embedded application during runtime.

Range

N/A.

See Also

GetAdministrationFile and ListNewGroups methods

Logging NNTP Property

Description

Indicates whether log events should be fired when log data is available.

Syntax

object.**Logging** [= *Boolean*]

Permission

W (Read/Write).

Availability

D (Design) and R(Runtime).

Data Type

Boolean.

Default Value

False.

Range

True or False

NotificationMode NNTP Property

Description

Determines when notification is issued for incoming data. Notification can also be suspended.

Syntax

object.**NotificationMode** [= *Integer*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

NotificationModeConstants.

Default Value

icContinuousMode.

Range

0-maximum unsigned long. At present, the values are:

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

OverviewSupported NNTP Property

Description

If True, the GetOverviewFormat and GetOverview methods may be used to retrieve header information stored in the server's overview database. This property has no meaning before the connection to the server has been established.

Syntax

object.**OverviewSupported**

Permission

R (Read-only).

Availability

R(Runtime).

Data Type

Boolean.

Default Value

False.

Range

True or False

Comments

Examine this property after a connection is established to determine if the server supports the OVERVIEW.FMT command.

PostingAllowed NNTP Property

Description

If True, the current NNTP server allows posting of news articles. This property has no meaning before the connection to the server has been established.

Syntax

object.**PostingAllowed**

Permission

R (Read-only).

Availability

R(Runtime).

Data Type

Boolean.

Default Value

True.

Range

True or False

Comments

Examine this property after a connection is established to determine if the server supports posting.

ProtocolState NNTP Property

Description

This property specifies the current state of the protocol.

Syntax

object.**ProtocolState**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

NNTPProtocolStateConstants.

Default Value

NNTPBase.

Range

0-1. Constants defined for the enum types of ProtocolState property are:

Value	Meaning
NNTPBase = 0	Base state before connection to server is established.
NNTPTransaction = 1	Connection to server is established. This is the valid state for calling methods on the control.

ProtocolStateString NNTP Property

Description

String representation of ProtocolState.

Syntax

object.**ProtocolStateString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

"BASE".

Range

N/A.

RemoteHost NNTP Property

Description

The remote machine to connect to if the RemoteHost parameter in the Connect method is missing. You can either provide a host name or an IP address string in dotted format. For example, “127.0.0.1”.

Syntax

object.**RemoteHost** [= *String*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

String.

Default Value

“news”.

Range

N/A.

RemotePort NNTP Property

Description

The remote port number to which to connect.

Syntax

object.**RemotePort** [= *Long*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

Long.

Default Value

119.

Range

1-65535.

ReplyCode NNTP Property

Description

The value of the reply code is a protocol specific number that determines the result of the last request, as returned in the ReplyString property.

Syntax

object.**ReplyCode**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

0

Range

See RFC 977 for a list of valid reply codes.

ReplyString NNTP Property

Description

Lists the last reply string sent by the NNTP Server to the client as a result of a request.

Syntax

object.**ReplyString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

SleepTime NNTP Property

Description

Specifies the sleep time between checking messages, if Blocking is True.

Syntax

object.**SleepTime** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

10 ms.

Range

>=zero.

Comments

Only applies when in Blocking mode.

SocketHandle NNTP Property

Description

Socket handle for the primary connection (Request/Reply connection).

Syntax

object.**SocketHandle**

Permission

R (Read only)

Availability

R (Runtime).

Data Type

Long.

Default Value

N/A.

Range

>=0

Comments

Some protocols require more than one connection. SocketHandle is the handle for the request/reply connection. If the SocketHandle is less than zero, the value is valid.

State NNTP Property

Description

This property specifies the connection state of the control.

Syntax

object.**State**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Integer.

Default Value

prcDisconnected.

Range

1-6. Constants defined for enum types of State property are:

Value	Meaning
prcConnecting = 1	Connecting. Connect has been requested, waiting for connect acknowledge.
prcResolvingHost = 2	Resolving Host. Occurs when RemoteHost is in name format rather than dot-delimited IP format.
prcHostResolved = 3	Resolved the host. Occurs only if ResolvingHost state has been entered previously.
prcConnected = 4	Connection established.
prcDisconnecting = 5	Connection closed. Disconnect has been initiated.
prcDisconnected = 6	Initial state when protocol object is instantiated, before Connect has been initiated, after a Connect attempt failed or after Disconnect performed.

StateString NNTP Property

Description

A string representation of State.

Syntax

object.**StateString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

"Disconnected".

Range

N/A.

Timeout NNTP Property

Description

Timeout value for the specified event. The event is specified by entering:

```
    Timeout(short event)
```

Syntax

object.**Timeout** (*event*) [= *Long*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

Long.

Default Value

0.

Range

0-maximum unsigned long. Constants defined for enum types for events are:

Value	Meaning
prcConnectTimeout = 1	Timeout for connect. If connection is not established within the timeout period, the Timeout event will be activated.
prcReceiveTimeout = 2	Timeout for receiving data. If no data arrives within the timeout period, the Timeout event will be activated.
prcUserTimeout= 65	Timeout for user defined event. Use prcUserTimeout + [Integer] range for custom timeout events.

URL NNTP Property

Description

URL string identifying the current document being transferred. The valid URL formats are:

```
      news:<newsgroupname>
      news<messageid>
```

Syntax

object.**URL** [= S*tring*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

String.

Default Value

Empty string.

Range

Valid URL.

Comments

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.

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 `nntp:` string may be omitted when using the NNTP control.

AboutBox NNTP Method

Description

Shows information about this control.

Return Value

Void.

Syntax

object.**AboutBox**

Parameters

None.

Cancel NNTP Method

Description

Cancels a pending request and returns the control to the Disconnected state.

Return Value

Void.

Syntax

object.**Cancel**

Parameters

None.

Connect NNTP Method

Description

Initiates a Connect request. The control calls the StateChanged and ProtocalStateChanged event if a connection is established.

Return Value

Void.

Syntax

object.**Connect** [*RemoteHost*], [*RemotePort*]

Parameters

RemoteHost

Optional. Remote host to which to connect. If this parameter is missing, the control connects to the host defined in the RemoteHost property.

Data Type: String

Param: IN

Default Value: N/A

RemotePort

Optional. Remote port to which to connect. If this parameter is missing, the control connects to the port defined in the RemotePort property.

Data Type: Long

Param: IN

Default Value: N/A

Comments

Optional arguments to this method override the values from corresponding RemoteHost and RemotePort properties. The values of the properties will not change. If no argument is given, the values from the properties will be used to establish the connection.

GetAdministrationFile NNTP Method

Description

Sends the NNTP XMOTD command to the server. This command retrieves the news server administrator's information if the information is newer than the value of lastUpdate.

Return Value

Void.

Syntax

object.**GetAdministrationFile** [*lastUpdate*]

Parameters

lastUpdate

Optional. Indicates time of last update from the server. If the lastUpdate argument is not given, the Control uses the value of the lastUpdate property.

Data Type: DATE

Param: IN

Default Value: None

GetArticleByArticleNumber NNTP Method

Description

Sends the NNTP ARTICLE command with articleNumber to the NNTP server. Upon successful completion, this method causes the DocOutput event to be activated.

Return Value

Void.

Syntax

object.**GetArticleByArticleNumber** [*articleNumber*]

Parameters

articleNumber

Optional. The article number of an article in the current newsgroup. The article number must be chosen from the range of articles numbers provided when the newsgroup was selected.   If it is omitted, the current article is assumed.

Data Type: VARIANT. Valid Variant types for articleNumber are String and Integer.

Param: IN

Default Value: None

GetArticleByMessageID NNTP Method

Description

Sends the NNTP ARTICLE command with articleID to the server. Upon successful completion, this method causes the DocOutput event to be activated.

Return Value

Void.

Syntax

object.**GetArticleByMessageID** *messageID*

Parameters

messageID

Specifies the article's unique messageID for the current NNTP server. The client may obtain the message-id from references contained within another article or from the message-id provided in the response to some other command.

Data Type: String

Param: IN

Default Value: None

GetArticleHeaders NNTP Method

Description

Sends the NNTP XHDR command to the server. Upon successful completion, this method causes the DocOutput event to be activated.

Return Value

Void.

Syntax

object.**GetArticleHeaders** *header*, [*firstArticle*], [*lastArticle*]

Parameters

header

The name of a header line (e.g.,"subject") in a news group article. This parameter is required. See RFC-1036 for a list of valid header lines.

Data Type: String

Param: IN

Default Value: None

firstArticle, lastArticle

Optional. If firstArticle and lastArticle are given, they indicate a range of article numbers. If lastArticle is 0, the range is all headers following firstArticle.

If no lastArticle argument is given, then the firstArticle indicates a message-id.

If neither firstArticle or lastArticle is given then information for the current article is retrieved.

Data Type: Long

Param: IN

Default Value: None

GetArticleNumbers NNTP Method

Description

Sends the NNTP command LISTGROUP to the server. Upon successful completion, this method causes the DocOutput event to be activated.

Return Value

Void.

Syntax

object.**GetArticleNumbers** [*groupName*]

Parameters

groupName

Optional. If the groupName argument is given, a list of article numbers for that group is retrieved and the group becomes the selected group. If the groupName argument is not given, a list of article numbers for the selected news group is retrieved.

Data Type: String.

Param: IN

Default Value: None

Comments

Use the ArticleNumbersSupported property after connection to determine if the current NNTP server supports this command.

GetBodyByArticleNumber NNTP Method

Description

Sends the NNTP BODY command with articleNumber to the NNTP server. Upon successful completion, this method causes the DocOutput event to be activated.

Return Value

Void.

Syntax

object.**GetBodyByArticleNumber** [*articleNumber*]

Parameters

articleNumber

Optional. The article number of an article in the current newsgroup. The article number must be chosen from the range of articles numbers provided when the newsgroup was selected.   If it is omitted, the current article is assumed.

Data Type: VARIANT. Valid Variant types for articleNumber are String and Integer.

Param: IN

Default Value: None

GetBodyByMessageID NNTP Method

Description

Sends the NNTP BODY command with messageID to the server. Upon successful completion, this method causes the DocOutput event to be activated.

Return Value

Void.

Syntax

object.**GetBodyByMessageID** *messageID*

Parameters

messageID

Specifies the article's unique messageID for the current NNTP server. The client will probably obtain the message-id from references contained within another article or from the message-id provided in the response to some other commands.

Data Type: String

Param: IN

Default Value: None

GetDoc NNTP Method

Description

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

Return Value

Void.

Syntax

object.**GetDoc** [*URL*], [*Headers*], [*OutputFile*]

Parameters

URL

Optional. The URL identifying the remote document to be retrieved.

Data Type: String

Param: IN

Default Value: DocOutput.URL

Headers

Optional. Headers used for requesting the document. This argument only applies to protocols where request headers can be specified (for example, HTTP).

Data Type: DocHeaders

Param: IN

Default Value: DocOutput.Headers

OutputFile

Optional. A local file to which the retrieved document will be written.

Data Type: String

Param: IN

Default Value: DocOutput.Filename

Comments

The GetDoc method in NNTP means retrieving an article from the NNTP 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) may be omitted and will default to the correct type for this control. For example, when using the NNTP control, the "nntp" string may 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.

For more information see DocInput and DocOutput Objects.

GetHeaderByArticleNumber NNTP Method

Description

Sends the NNTP HEAD command with articleNumber to the NNTP server. Upon successful completion, this method causes the DocOutput event to be activated.

Return Value

Void.

Syntax

object.**GetHeaderByArticleNumber** [*articleNumber*]

Parameters

articleNumber

Optional. The article number of an article in the current newsgroup. The article number must be chosen from the range of articles numbers provided when the newsgroup was selected.   If it is omitted, the current article is assumed.

Data Type: VARIANT. Valid Variant types for articleNumber are String and Integer.

Param: IN

Default Value: None

GetHeaderByMessageID NNTP Method

Description

Sends the NNTP HEAD command with messageID to the server. Upon successful completion, this method causes the DocOutput event to be activated.

Return Value

Void.

Syntax

object.**GetHeaderByMessageID** *messageID*

Parameters

messageID

Specifies the article's unique messageID for the current NNTP server. The client will probably obtain the message-id from references contained within another article or from the message-id provided in the response to some other commands.

Data Type: String

Param: IN

Default Value: None

GetOverview NNTP Method

Description

Sends the XOVER command to the server. Use the OverSupported property after connection to determine if the current NNTP server supports this command. When this method reaches a successful completion, the DocInput event is activated.

Return Value

Void.

Syntax

object.**GetOverview** [*firstArticle*], [*lastArticle*]

Parameters

firstArticle, lastArticle

Optional. If firstArticle and lastArticle are given, they indicate a range of article numbers. If lastArticle is 0, the range is all headers following firstArticle.

If neither firstArticle or lastArticle is given then information for the current article is retrieved.

Data Type: String.

Param: IN

Default Value: N/A

Comment

The XOVER command returns information from the overview database for the article(s) specified.

GetOverviewFormat NNTP Method

Description

Sends the LIST OVERVIEW.FMT command to the server. Use the OverViewSupported property after connection to determine if the current NNTP server supports this command. When this method reaches a successful completion, the DocInput event is activated.

Return Value

Void.

Syntax

object.**GetOverviewFormat**

Parameters

None.

Comments

The LIST OVERVIEW.FMT command is used to retrieve a list of headers in the order they appear in the servers overview database.

GetStatByArticleNumber NNTP Method

Description

Sends the NNTP STAT command with articleNumber to the NNTP server. When this method reaches a successful completion, the StatArticle event is activated.

Return Value

Void.

Syntax

object.**GetStatByArticleNumber** [*articleNumber*]

Parameters

articleNumber

Optional. The article number of an article in the current newsgroup. The article number must be chosen from the range of articles numbers provided when the newsgroup was selected.   If it is omitted, the current article is assumed.

Data Type: VARIANT. Valid Variant types for articleNumber are String and Integer.

Param: IN

Default Value: None

ListGroups NNTP Method

Description

Sends NNTP LIST command to the server. The server responds with a list of all news groups. Upon successful completion, this method causes the DocOutput event to be activated.

Return Value

Void.

Syntax

object.**ListGroups**

Parameters

None.

ListGroupDescriptions NNTP Method

Description

Sends the NNTP LIST NEWSGROUPS command to the server. Upon successful completion, this method causes the DocOutput event to be activated.

Return Value

Void.

Syntax

object.**ListGroupDescriptions**

Parameters

None.

ListNewGroups NNTP Method

Description

Sends NNTP NEWGROUPS command to server. When this method reaches a successful completion, the DocInput event is activated.

Return Value

Void.

Syntax

object.**ListNewGroups** [*lastTime*]

Parameters

lastTime

Optional. Indicates the last time articles were retrieved by the client. If the lastTime parameter is not given, the Control uses the value of the lastUpdate property.

Data Type: DATE.

Param: IN

Default Value: lastUpdate property

Quit NNTP Method

Description

Sends NNTP QUIT command and disconnects from the NNTP server. When this method reaches a successful completion, the StateChanged event is activated.

Return Value

Void.

Syntax

object.**Quit**

Parameters

None.

SelectGroup NNTP Method

Description

Sends NNTP GROUP command to the server. On successful completion, the SelectGroup event is activated.

Return Value

Void.

Syntax

object.**SelectGroup** *groupName*

Parameters

groupName

The name of the group of articles to be selected.

Data Type: String.

Param: IN

Default Value: None

SendDoc NNTP Method

Description

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

Return Value

Void.

Syntax

object.**SendDoc** [*URL*], [*Headers*], [*InputData*], [*InputFile*], [*OutputFile*]

Parameters

URL

Optional. The URL identifying the remote document to be sent. If specified, the URL property will be set to this value.

Data Type: String

Param: IN

Default Value: DocInput.URL

Headers

Optional. Headers used for sending the document. The user should modify the values in DocInput.Headers directory.

Data Type: DocHeaders

Param: IN

Default Value: DocInput.Headers

InputData

Optional. A data buffer containing the document to be sent.

Data Type: VARIANT

Param: IN

Default Value: DocInput.GetData

InputFile

Optional. A local file containing the document to be sent.

Data Type: String

Param: IN

Default Value: DocInput.Filename

OutputFile

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).

Data Type: String

Param: IN

Default Value: DocOutput.Filename

Comments

The SendDoc method in NNTP means posting an article 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) may be omitted and will default to the correct type for this control. For example, when using the NNTP control, the "nntp:" string may 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.

For more information see DocInput and DocOutput Objects.

SetLastArticle NNTP Method

Description

Sends NNTP LAST command to the server. On successful completion, the LastArticle event is activated.

Return Value

Void.

Syntax

object.**SetLastArticle**

Parameters

None.

SetNextArticle NNTP Method

Description

Sends NNTP NEXT command to the server. On successful completion, the NextArticle event is activated.

Return Value

Void.

Syntax

object.**SetNextArticle**

Parameters

None.

NNTP Events

Events are used for NNTP client notification. They indicate that an action has been requested and processed. Any errors which occur during command processing result in the Error event being called with appropriate error codes. Error codes, state changes, and protocol return values are usually checked during event processing.

The following series of Help topics describe the events supported by the NNTP Client Control. Each description includes the syntax, related parameters, their data type, default value, and whether the parameter is used for input or output (IN or OUT). For a complete listing of events, see NNTP Client ActiveX Control Overview.

ArticleStatus NNTP Event

Description

This event is activated after a successful completion of the GetStatByArticleNumber method.

Syntax

*object*_**ArticleStatus** (*ArticleNumber* **As Long,** *MessageID* **As String**)

Parameters

ArticleNumber

The article number of the selected article.

Data Type: Long

Param: IN

Default Value: N/A

MessageID

The message id of the selected article.

Data Type: String

Param: IN

Default Value: N/A

AuthenticateRequest NNTP Event

Description

This event is activated when the connected NNTP server requests authentication.

Syntax

*object*_**AuthenticateRequest** (*UserID* **As String,** *Password* **As String**)

Parameters

UserId

Optional. User identification string to use for authentication.

Data Type: String

Param: IN

Default Value: N/A

Password

Optional. Password to use for authentication.

Data Type: String

Param: IN

Default Value: N/A

Comments

If the UserID and Password arguments are specified, their values are used instead of the UserID and Password properties.

AuthenticateResponse NNTP Event

Description

This event is activated when an authentication response is received from the server.

Syntax

*object*_**AuthenticateResponse** (*Authenticated* **As Boolean**)

Parameters

Authenticated

Indicates if the authentication is successful. If this argument is True, the authentication has succeeded.

Data Type: Boolean

Banner NNTP Event

Description

This event is activated when the server responds with its sign-on banner after a connection is established.

Syntax

*object*_**Banne**r (*Banner* **As String**)

Parameters

Banner

The sign-on message returned by the news server.

Data Type: String

Param: IN

Default Value: N/A

Busy NNTP Event

Description

This event is activated when a command is in progress or when a command has completed.

Syntax

*object*_**Busy** (*Busy* **As Boolean**)

Parameters

Busy

Indicates whether or not a command is in progress.

Data Type: Boolean. If the argument is True, a command is in progress.

Cancel NNTP Event

Description

This event is activated after a cancellation request has been completed and satisfied. After this event the object's state changes to Disconnected.

Syntax

*object*_**Cancel**

Parameters

None.

DocInput NNTP Event

Description

A DocInput related event that indicates the input data has been transferred or the DocInput state has changed.

Syntax

*object*_**DocInput** (*DocInput* **As DocInput**)

Parameters

DocInput

Object describing document input data for the current transfer.

Data Type: DocInput

Param: IN

Default Value: N/A

Comments

The DocInput event can be used in its basic form for notification of transfer progress, (for example, for updating a progress bar). The DocInput.BytesTotal, DocInput.BytesTransferred and DocInput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

To gain more power from this control, you can also use the DocInput event for data streaming. For more information, see DocInput Object Overview.

DocOutput NNTP Event

Description

A DocOutput related event indicating that output data has been transferred or the DocOutput state has changed.

Syntax

*object*_**DocOutput** (*DocOutput* **As DocOutput**)

Parameters

DocOutput

Object describing document output data for the current transfer.

Data Type: DocOutput

Param: IN

Default Value: N/A

Comments

The DocOutput event can be used in its basic form to notify the user of transfer progress, (for example, for updating a progress bar). The DocOutput.BytesTotal, DocOutput.BytesTransferred and DocOutput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

To gain more power from this control, you can also use the DocOutput event for data streaming. For more information, see DocInput Object Overview.

Error NNTP Event

Description

This event is activated when an error occurs in background processing (for example, failed to connect or failed to send or receive in the background).

Syntax

*object*_**Error** (*ErrCode* **As Integer**, *Description* **As String**, *Scode* **As Long**, *Source* **As String**, *HelpFile* **As String**, *HelpContext* **As Long**, *CancelDisplay* **As Boolean**)

Parameters

ErrCode

The short error code. For a list of possible NNTP error codes see NNTP Error Codes.

Description

String containing error information.

sCode

The long Scode.

Source

Error source.

HelpFile

Help file name.

HelpContext

Help file context.

CancelDisplay

Indicates whether to cancel the display. The default is TRUE (no display of the default error message box). If you do want to use the default message box, set CancelDisplay to FALSE.

LastArticle NNTP Event

Description

This event is activated after a successful completion of the SetLastArticle method.

Syntax

*object*_**LastArticle** (*ArticleNumber* **As Long,** *MessageID* **As String**)

Parameters

ArticleNumber

The article number of the selected article.

Data Type: Long

Param: IN

Default Value: N/A

MessageID

The message id of the selected article.

Data Type: String

Param: IN

Default Value: N/A

Log NNTP Event

Description

This event is fired when logging data is available.

Syntax

*object*_**Log**

Parameters

None.

NextArticle NNTP Event

Description

This event is activated after a successful completion of the SetNextArticle method.

Syntax

*object*_**NextArticle** (*ArticleNumber* **As Long,** *MessageID* **As String**)

Parameters

ArticleNumber

The article number of the selected article.

Data Type: Long

Param: IN

Default Value: N/A

MessageID

The message id of the selected article.

Data Type: String

Param: IN

Default Value: N/A

ProtocolStateChanged NNTP Event

Description

This event is activated whenever the protocol state changes.

Syntax

*object*_**ProtocolStateChanged** (*ProtocolState* **As Integer**)

Parameters

Refer to the ProtocolState property and ProtocolStateString property for possible values of the ProtocolState parameter.

SelectGroup NNTP Event

Description

This event is activated after a successful completion of the SelectGroup method.

Syntax

*object*_(*groupName* **As String,** *firstArticleNumber* **As Long,** *lastArticleNumber* **As Long**, *msgCount* **As Long**

Parameters

groupName

The name of the group of articles to be selected.

Data Type: String

firstArticleNumber

The number of the first article in the selected news group.

Data Type: Long

lastArticleNumber

The number of the last article in the selected news group.

Data Type: Long

msgCount

An estimate, provided by the NNTP server, of the number of articles in the group.

Data Type: Long

StateChanged NNTP Event

Description

This event is activated whenever the state of the transport state changes.

Syntax

*object*_**StateChanged** (*State* **As Integer**)

Parameters

Refer to the State property and StateString for possible values of the state parameter.

TimeOut NNTP Event

Description

This event is activated when the timer for the specified event expires.

Syntax

*object*_**TimeOut** (ByVal *Event* **As Integer,** *Continue* **As Boolean**)

Parameters

Event

Defines the event to which the time interval applies.

Data Type: Short

Continue

Determines if the timer is active or not. Set *Continue* to TRUE to keep the timer active.

Data Type: Boolean

Default Value: False

Comments

Event is an integer value that determines the type of Timeout event that will be enabled. Constants defined for enum types for events are:

Value	**Meaning**
prcConnectTimeout = 1	Timeout for connect. If connection is not established within the timeout period, the Timeout event will be activated.
prcReceiveTimeout = 2	Timeout for receiving data. If no data arrives within the timeout period, the Timeout event will be activated.
prcUserTimeout= 65	Timeout for user defined event. Use prcUserTimeout + [Integer] range for custom timeout events.

{button ,JI(`NIA.HLP',`IDH_Timeout')} **See Also**

NNTP Sample Session

A typical use of the NNTP Control might be to develop a custom news reader. A code fragment from a simple application using the NNTP Control to post a news article would look similar to this:

```
DocInput.headers.Add “From”, “joe@xyz.com”
DocInput.headers.Add “Path”, “joe@abc.com”
DocInput.headers.Add “Newsgroups”, “comp.lang.c++”
DocInput.headers.Add “Subject”, “C++ vs. Java”
DocInput.headers.Add “Message-ID”, “<12345joe@xyz.com”
DocInput.headers.Add “Date”, “Jan 24, 1996”

Dim articleText as String
articleText = “Is Java better than C++?”
Nntpct1.RemoteHost = “news”
Nntpct1.SendDoc,,articleText
```

POP Client ActiveX Control Overview

The POP Client Control provides access to Internet mail servers using the POP3 protocol, as specified by RFC 1081, *Post Office Protocol*. It can be used by Internet mail developers or system integrators. The major advantage of this control is its ability to retrieve mail from UNIX or other servers supporting POP3 protocol.

The main features of the POP ActiveX Control are that it:

- Connects to a server
- Sends authentication information (user and password) to the server
- Retrieves user mailbox information, such as the number of messages waiting to be retrieved
- Retrieves messages from the server
- Deletes messages from the server

The following table lists the properties, methods and events supported by the POP Client Control. For an example illustrating the use of the control in a real life situation, see POP Sample Session.

Property	Method	Event
Blocking	AboutBox	Busy
BlockResult	Authenticate	Cancel
Busy	Cancel	Delete
DocOutput	Connect	DocOutput
EnableTimer	Delete	Error
Errors	GetDoc	Last
LocalPort	Last	Log
Logging	MessageSize	MessageSize
MessageCount	NOOP	NOOP
NotificationMode	Quit	ProtocolStateChanged
Password	RefreshMessageCount	RefreshMessageCount
ProtocolState	Reset	Reset
ProtocolStateString	RetrieveMessage	StateChanged
ReplyCode	TopMessage	TimeOut
ReplyString		
RemoteHost		
RemotePort		
SleepTime		
SocketHandle		
State		
StateString		
Timeout		
TopLines		
TopSupported		
URL		
UserID		

POP3 Commands

The following table summarizes POP3 commands as described by the protocol specification in RFC 1081. All the commands are implemented in the control, although some of them are abstracted at a higher level (e.g. USER + PASS = Authorization).

Command	Usage	When Valid
DELE msg	required	TRANSACTION state
LAST	required	TRANSACTION state

LIST [msg]	required	TRANSACTION state
NOOP	required	TRANSACTION state
PASS string	required	AUTHORIZATION state
QUIT	required	AUTHORIZATION and UPDATE state
RETR msg	required	TRANSACTION state
RPOP user	optional	AUTHORIZATION state; not supported in current release
RSET	required	TRANSACTION state
STAT	required	TRANSACTION state
TOP msg n	optional	TRANSACTION state; supported if it is supported by the server.
USER name	required	AUTHORIZATION state

Each of the POP3 commands can return either:

- +OK
- -ERR

Note: The reply given by the POP3 server to any command is significant only up to "+OK" and "-ERR". The client can ignore any text occurring after this reply. The only exception is the STAT command.

Using the POP Client Control

To use the POP Client ActiveX Control you must choose the POP toolbox icon.

There should be no speed overhead and response delay other than the one given by the network. This control uses and is dependent on the DocStream objects (DocInput and DocOutput).

Blocking POP Client Property

Description

Indicates whether methods should block until complete or not.

Syntax

object.**Blocking** *[=Boolean]*

Permission

W (Read/Write)

Availability

D (Design) and R (Runtime)

Data Type

Boolean.

Default Value

False.

Range

True or False.

Comments

If Blocking is set to True, then a method will not be completed until the proper event(s) for a given request are fired, if applicable.

BlockResult POP Client Property

Description

Returns the result value of the last blocking method called.

Syntax

object.**BlockResult**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

BlockingResultConstants.

Default Value

icBlockOK..

Range

Name	Value	Description
icBlockOK	0	Blocking method was successful.
IcTimedOut	1	Blocking method returned due to timeout.
IcErrorExit	2	Blocking method returned due to an error.
IcBlockCancel	3	Blocking method returned due to cancel.
IcUserQuit	4	Blocking method returned due application end.

Busy POP Client Property

Description

True indicates indicates that a command is in progress, and False indicates that a command has completed.

Syntax

object.**Busy** [= *Boolean*]

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Boolean.

Default Value

N/A.

Range

N/A.

DocOutput POP Client Property

Description

Object describing output information for the document being transferred.

Syntax

object.**DocOutput**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

DocOutput.

Default Value

N/A.

Range

N/A.

Comments

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

Properties of the DocOutput object may be set before calling the GetDoc method or they may be passed as arguments to this method. The DocOutput object is also used for conveying information about the progress of the document transfer, and for data linking and streaming.

For more information, see DocOutput POP Client Event the DocOutput event Common Control Objects.

EnableTimer POP Client Property

Description

Enable timer for the specified event. The event is specified by entering:

```
    EnableTimer(short event)
```

Syntax

object.**EnableTimer** *(event)* [= *Boolean*]

Permission

W (Write Only).

Note: This is the only control property that is Write only.

Availability

R (Runtime)

Data Type

Boolean.

Default Value

False. (The timer for this event will not be enabled.)

Range

True or False

Comments

Event is an integer value that determines the type of Timeout event that will be enabled. Constants defined for enum types for events are:

Value	Meaning
prcConnectTimeout = 1	Timeout for connect. If connection is not established within the timeout period, the Timeout event will be activated.
prcReceiveTimeout = 2	Timeout for receiving data. If no data arrives within the timeout period, the Timeout event will be activated.
prcUserTimeout= 65	Timeout for user defined event. Use prcUserTimeout + [Integer] range for custom timeout events.

Errors POP Client Property

Description

A collection of errors that can be accessed for details about the last error that occurred. This collection should be used within an Error event if information passed through the Error event is not sufficient. For more details, see icErrors Item Overview.

Syntax

object.**Errors**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

icErrors.

Default Value

N/A.

Range

N/A.

LocalPort POP Client Property

Description

Designates the local port to use.

Syntax

object.**LocalPort** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

0.

Range

0 - 65535

Logging POP Client Property

Description

Indicates whether log events should be fired when log data is available.

Syntax

object.**Logging** [= *Boolean*]

Permission

W (Read/Write).

Availability

D (Design) and R(Runtime).

Data Type

Boolean.

Default Value

False.

Range

True or False

MessageCount POP Client Property

Description

This property specifies the number of messages in the mailbox. It is established after authentication has been successfully performed. Before that, it is invalid.

Syntax

object.**MessageCount**

Permission

R (Read-only).

Availability

R (Runtime)

Data Type

Integer.

Default Value

0.

Range

1-32767.

NotificationMode POP Client Property

Description

Determines when notification is issued for incoming data. Notification can also be suspended.

Syntax

object.**NotificationMode** [= *Integer*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

NotificationModeConstants.

Default Value

icContinuousMode.

Range

0-maximum unsigned long. At present, the values are:

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

Password POP Client Property

Description

Password of current user on the server.

Syntax

object.**Password** [= *String*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

String.

Default Value

Empty.

Range

N/A.

ProtocolState POP Client Property

Description

This property specifies the current state of the protocol.

Syntax

object.**ProtocolState**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

popProtocolSateConstants.

Default Value

popBase.

Range

0-3. Constants defined for enumerated type Protocol State property:

Constant	Meaning
popBase = 0	Base state before connection to server is established.
popAuthorization = 1	Authorization is being performed.
popTransaction = 2	Authorization had been performed successfully, the client has successfully identified itself to the POP3 server and the POP3 server has locked and burst the appropriate maildrop.
popUpdate = 3	When Quit command is issued from transaction state.

ProtocolStateString POP Client Property

Description

String representation of ProtocolState.

Syntax

object.**ProtocolStateString** [= String]

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

“BASE”.

Range

N/A.

RemoteHost POP Client Property

Description

The remote machine to connect to if the RemoteHost parameter in the Connect method is missing. You can either provide a host name or an IP address string in dotted format. For example, 127.0.0.1.

Syntax

object.**RemoteHost** [= *String*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

String.

Default Value

“127.0.0.1”.

Range

N/A.

RemotePort POP Client Property

Description

The remote port number to which to connect.

Syntax

object.**RemotePort** [= *Long*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

Long.

Default Value

110.

Range

1-65535.

ReplyCode POP Client Property

Description

The value of the reply code is a protocol specific number that determines the result of the last request, as returned in the ReplyString property.

Syntax

object.**ReplyCode**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

0

Range

See RFC 1081 for a list of valid reply codes.

ReplyString POP Client Property

Description

Line returned to the client as a result of a request.

Syntax

object.**ReplyString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

SleepTime POP Client Property

Description

Specifies the sleep time between checking messages, if Blocking is True.

Syntax

object.**SleepTime** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

10 ms.

Range

>=zero.

Comments

Only applies when in Blocking mode.

SocketHandle POP Client Property

Description

Socket handle for the primary connection (Request/Reply connection).

Syntax

object.**SocketHandle**

Permission

R (Read only)

Availability

R (Runtime).

Data Type

Long.

Default Value

N/A.

Range

>=0

Comments

Some protocols require more than one connection. SocketHandle is the handle for the request/reply connection. A value less than zero indicates that the Sockethandle is not available.

State POP Client Property

Description

This property specifies the connection state of the control.

Syntax

object.**State**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Integer.

Default Value

prcDisconnected.

Range

1-6. Constants defined for enum types of State property are:

Value	Meaning
prcConnecting = 1	Connecting. Connect has been requested, waiting for connect acknowledge.
prcResolvingHost = 2	Resolving Host. Occurs when RemoteHost is in name format rather than dot-delimited IP format.
prcHostResolved = 3	Resolved the host. Occurs only if ResolvingHost state has been entered previously.
prcConnected = 4	Connection established.
prcDisconnecting = 5	Connection closed. Disconnect has been initiated.
prcDisconnected = 6	Initial state when protocol object is instantiated, before Connect has been initiated, after a Connect attempt failed or after Disconnect performed.

StateString POP Client Property

Description

A string representation of State.

Syntax

object.**StateString** [= *String*]

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

"Disconnected".

Range

N/A.

Timeout POP Client Property

Description

Timeout value for the specified event. The event is specified by entering:

```
Timeout(short event)
```

Syntax

object.**Timeout** (*event)* [= *Long*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

Long.

Default Value

0.

Range

0-maximum unsigned long. Constants defined for enum types for events are:

Value	Meaning
prcConnectTimeout = 1	Timeout for connect. If connection is not established within the timeout period, the Timeout event will be activated.
prcReceiveTimeout = 2	Timeout for receiving data. If no data arrives within the timeout period, the Timeout event will be activated.
prcUserTimeout= 65	Timeout for user defined event. Use prcUserTimeout + [Integer] range for custom timeout events.

TopLines POP Client Property

Description

Designates the number of lines to be retrieved in a top request.

Syntax

object.**TopLines** [= *Long*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

Long.

Default Value

0.

Range

0-2147483647.

TopSupported POP Client Property

Description

This property indicates “Top is supported.” It can be queried after a connection to the server has been established. It is set to TRUE if the particular server supports the TOP command.

Syntax

object.**TopSupported**

Permission

R (Read-only).

Availability

R (Runtime)

Data Type

Boolean.

Default Value

None.

Range

True or False.

URL POP Client Property

Description

URL string identifying the current document being transferred. The URL format for this control is:

`POP://user:password@host:`*`port/message number`*

Syntax

object.**URL** [= S*tring*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

String.

Default Value

Empty string.

Range

Valid URL.

Comments

URL may be set before calling the GetDoc 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 POP control, the URL property may identify a message being retrieved from a remote server. 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 pop string may be omitted when using the POP control.

UserId POP Client Property

Description

User identification name for the client on the server.

Syntax

object.**UserId** [= *String*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

String.

Default Value

Empty.

Range

N/A.

POP Client Methods

Methods are called to perform a particular operation on an object. After the method is successfully processed, you will receive an event with a name similar to the method called. You can then check the ReplyString value for the server response or check error codes if an error message is generated.

The following series of Help topics describe the methods supported by the POP Client Control. For an explanation of the description categories, see Object Descriptions,

AboutBox POP Client Method

Description

Shows information about this control.

Return Value

Void.

Syntax

object.**AboutBox**

Parameters

None.

Authenticate POP Client Method

Description

Authenticates the user based on the parameters passed. If no parameters are passed, the UserId and Password properties are used. If neither the UserId nor the Password are entered, the control uses the URL. When authentication process is terminated, the Authenticate event is activated.

Return Value

Void.

Syntax

object.**Authenticate** [*UserID],* [*Password*]

Parameters

UserId

Optional. User identification string to use for authentication.

Data Type: String

Param: IN

Default Value: N/A

Password

Optional. Password to use for authentication.

Data Type: String

Param: IN

Default Value: N/A

Comments

If the UserId and/or Password are set before invoking this method, the optional parameters do not need to be specified. Optional arguments to this method override the values from corresponding UserId and Password properties. The values of the properties will change. If you omit one or both of the arguments, the value from a corresponding property will be used to provide authentication.

Cancel POP Client Method

Description

Cancels a pending request and returns the state to Disconnected.

Return Value

Void.

Syntax

object.**Cancel**

Parameters

None.

Connect POP Client Method

Description

Initiates a Connect request. The control calls the StateChanged event if a connection is established.

Return Value

Void.

Syntax

object.**Connect** [*RemoteHost*], [*RemotePort*]

Parameters

RemoteHost

Optional. Remote host to which to connect. If this parameter is missing, the control connects to the host defined in the RemoteHost property.

Data Type: String

Param: IN

Default Value: N/A

RemotePort

Optional. Remote port to which to connect. If this parameter is missing, the control connects to the port defined in the RemotePort property.

Data Type: Long

Param: IN

Default Value: N/A

Comments

Optional arguments to this method override the values from corresponding RemoteHost and RemotePort properties. The values of the properties will change. If no argument is given, the values from the properties will be used to establish the connection.

Delete POP Client Method

Description

Initiates a Delete request. If successful, a Delete event is activated, otherwise an Error event is activated.

Return Value

Void.

Syntax

object.**Delete** *MsgNumber*

Parameters

MsgNumber

Number of message to be deleted.

Data Type: Integer.

Param: IN

Default Value: None.

GetDoc POP Client Method

Description

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

Return Value

Void.

Syntax

object.**GetDoc** [*URL*], [*Headers*], [*OutputFile*]

Parameters

URL

Optional. The URL identifying the remote document to be retrieved.

Data Type: String

Param: IN

Default Value: DocInput.URL

Headers

Optional. Headers used for requesting the document. This argument only applies to protocols where request headers can be specified (for example, HTTP).

Data Type: DocHeaders

Param: IN

Default Value: DocInput.Headers

OutputFile

Optional. A local file to which the retrieved document will be written.

Data Type: String

Param: IN

Default Value: DocOutput.Filename

Comments

The GetDoc method in POP gets a message 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) may be omitted and will default to the correct type for this control. For example, when using the POP control, the "pop:" string may 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.

For more information see DocInput and DocOutput Objects.

Last POP Client Method

Description

Initiates a LAST request. If successful, the LAST event is activated. This request is used to find the highest message number accessed by the client.

Return Value

Void.

Syntax

object.**Last**

Parameters

None.

MessageSize POP Client Method

Description

Initiates a request to retrieve the message size. If successful, a MessageSize event is activated, otherwise the Error event is activated.

Return Value

Void.

Syntax

object.**MessageSize** *MsgNumber*

Parameters

MsgNumber

Message number.

Data Type: Integer.

Param: IN

Default Value: None.

NOOP POP Client Method

Description

Initiates a NOOP request. This is used to test the connection.

Return Value

Void.

Syntax

object.**NOOP**

Parameters

None.

Quit POP Client Method

Description

Initiates a Quit request. If unsuccessful, the Error event is activated.

Return Value

Void.

Syntax

object.**Quit**

Parameters

None.

RefreshMessageCount POP Client Method

Description

This method will refresh the number of undeleted messages from your current maildrop. When the request is completed, the RefreshMessageCount event is activated, indicating the current number of undeleted messages. This method is only available if you are already connected and authenticated. You will not be notify you of new messages received by the POP server; the context is the current maildrop. The message numbers themselves will not be renumbered. If, for example, you delete message 4 out of 6, a message retrieval of 5 will get message 5 and not message 6; message 6 is not renumbered to 5. Using Reset, you can undo message deletion at anytime prior to a Quit command. If, for example, you delete messages 3 and 5 out of 6, and then call RefreshMessageCount, the new number of undeleted messages is 4. If you do a Reset and then call RefreshMessageCount, the number of messages is 6.

Return Value

Void.

Syntax

object.**RefreshMessageCount**

Parameters

None.

Reset POP Client Method

Description

Initiates a RSET request. Any messages marked as deleted will be unmarked. If successful, a corresponding Reset event is activated, otherwise an Error event is activated.

Return Value

Void.

Syntax

object.**Reset**

Parameters

None.

RetrieveMessage POP Client Method

Description

Initiates a RetrieveMessage request for the message specified in msgNumber.

Return Value

Void.

Syntax

object.**RetrieveMessage** *msgNumber*

Parameters

msgNumber

specifies number of message to be retrieved.

Data Type: Integer.

Param: IN

Default Value: None.

Comments

DocOutput event can be used to retrieve the data.

TopMessage POP Client Method

Description

Initiates a Top of Message request for the message specified in msgNumber.

Syntax

object.**TopMessage** *msgNumber*

Parameters

msgNumber

specifies number of message to be retrieved.

Data Type: Integer.

Param: IN

Default Value: None.

Comments

DocOutput event can be used to retrieve the data.

TopMessage is used in conjunction with the TopLines property. If TopLines is 0, then only header information will be retrieved.

POP Client Events

Events are used for POP client notification. They indicate that an action has been requested and processed. Any errors which occur during command processing result in the Error event being called with appropriate error codes. Error codes (see icError Item Overview), state changes, and protocol return values are usually checked during event processing.

The following Help topics describe the events supported by the POP Client Control. Each description includes the syntax, related parameters, their data type, default value, and whether the parameter is used for input or output (IN or OUT). For a complete listing of POP events, see POP Client ActiveX Control Overview.

Busy POP Client Event

Description

This event is activated when a command is in progress or when a command has completed.

Syntax

*object*_**Busy** (*Busy* **As Boolean**)

Parameters

Busy

Indicates whether or not a command is in progress.

Data Type: Boolean. If the argument is True, a command is in progress.

Cancel POP Client Event

Description

This event is activated after a cancellation request has been completed and satisfied. After this event the object's state changes to prcDisconnected.

Syntax

*object*_**Cancel**

Parameters

None.

Delete POP Client Event

Description

This event is activated after the successful completion of a Delete request.

Syntax

*object*_**Delete**

Parameters

None.

DocOutput POP Client Event

Description

A DocOutput related event indicating that output data has been transferred or the DocOutput state has changed.

Syntax

*object*_**DocOutput** (*DocOutput* **As DocOutput**)

Parameters

DocOutput

Object describing document output data for the current transfer.

Data Type: DocOutput

Param: IN

Default Value: N/A

Comments

The DocOutput event can be used in its basic form to notify the user of transfer progress, (for example, for updating a progress bar). The DocOutput.BytesTotal, DocOutput.BytesTransferred and DocOutput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

To gain more power from this control, you can also use the DocOutput event for data streaming. For more information, see DocInput Object Overview.

Error POP Client Event

Description

This event is activated when an error occurs in background processing (for example, failed to connect or failed to send or receive in the background).

Syntax

*object*_**Error** (*ErrCode* **As Integer**, *Description* **As String**, *Scode* **As Long**, *Source* **As String**, *HelpFile* **As String**, *HelpContext* **As Long**, *CancelDisplay* **As Boolean**)

Parameters

ErrCode

The short error code. For a list of possible POP error codes, see POP Error Codes.

Description

String containing error information.

sCode

The long Scode.

Source

Error source.

HelpFile

Help file name.

HelpContext

Help file context.

CancelDisplay

Indicates whether to cancel the display. The default is TRUE (no display of the default error message box). If you do want to use the default message box, set CancelDisplay to FALSE.

Last POP Client Event

Description

This event is activated after the successful completion of a Last request. It indicates the number of the last message accessed by the client.

Syntax

*object*_**Last** *(Number* **As Long***)*

Parameters

Number

Number of the last message accessed by the client.

Data Type: Long.

Log POP Client Event

Description

This event is fired when logging data is available.

Syntax

*object*_**Log**

Parameters

None.

MessageSize POP Client Event

Description

This event is activated after successful completion of a MessageSize request.

Syntax

*object*_**MessageSize** *(MsgSize* **As Long***)*

Parameters

MsgSize

Size of the message requested.

Data Type: Long.

NOOP POP Client Event

Description

This event is activated after the successful completion of a NOOP request.

Syntax

*object*_**NOOP**

Parameters

None.

ProtocolStateChanged POP Client Event

Description

This event is activated whenever the protocol state changes.

Syntax

*object*_**ProtocolStateChanged** (*ProtocolState* **As Integer**)

Parameters

Refer to the ProtocolState property and ProtocolStateString for possible values of the ProtocolState parameter.

RefreshMessageCount POP Client Event

Description

This event is activated after a successful completion of RefreshMessageCount request. The number of undeleted messages from the current maildrop is returned. (A maildrop contains the messages that can be retrieved/deleted in the current state.)

Syntax

*object*_**RefreshMessageCount**

Parameters

None.

Reset POP Client Event

Description

This event is activated after the successful completion of a Reset request.

Syntax

*object*_**Reset**

Parameters

None.

StateChanged POP Client Event

Description

This event is activated whenever the state of the transport state changes.

Syntax

*object*_**StateChanged** (*State* **As Integer)**

Parameters

Refer to the State property and StateString for possible values of the state parameter.

TimeOut POP Client Event

Description

This event is activated when the timer for the specified event expires.

Syntax

*object*_**TimeOut** (ByVal *Event* **As Integer,** *Continue* **As Boolean**)

Parameters

Event

Defines the event to which the time interval applies.

Data Type: Short

Continue

Determines if the timer is active or not. Set *Continue* to TRUE to keep the timer active.

Data Type: Boolean

Default Value: False

Comments

Event is an integer value that determines the type of Timeout event that will be enabled. Constants defined for enum types for events are:

Value	**Meaning**
prcConnectTimeout = 1	Timeout for connect. If connection is not established within the timeout period, the Timeout event will be activated.
prcReceiveTimeout = 2	Timeout for receiving data. If no data arrives within the timeout period, the Timeout event will be activated.
prcUserTimeout= 65	Timeout for user defined event. Use prcUserTimeout + [Integer] range for custom timeout events.

POP Sample Session

Click on one of the following POP sample sessions, which demonstrates how to use the POP ActiveX Control. An application may want to retrieve the message for user `johng` whose password is `secret` on the Mail server. To achieve this a sequence of methods is chained through events in the VB program as shown in Connecting.

Sample Sessions

{button ,JI(`NIA.HLP',`IDH_Connecting')} **Connecting**

{button ,JI(`NIA.HLP',`IDH_Authentication')} **Authentication**

{button ,JI(`NIA.HLP',`IDH_Retrieval')} **Retrieval**

{button ,JI(`NIA.HLP',`IDH_Receiving_a_Reply_POP')} **Receiving a Reply**

Connecting

The Connect method is called to connect.

```
Private Sub btnConnect_Click(Index As Integer)
Popct1.RemoteHost = rhost.Text
Popct1.Connect
End Sub
```

Authentication

The Authenticate method is called once the connection is established to send authentication information to the server.

```
Private Sub Command1_Click()
‘ authenticate the user
‘ UserId and Password properties are used
Popct1.UserId = UserId.Text
Popct1.Password = Password.Text
Popct1.Authenticate
End Sub
```

Retrieval

At this point the total number of messages residing on the server for this user is known and can be retrieved.

```
Private Sub Command3_Click()
msgTxt = ""
Popct1.MessageRetrieve Val(txtMsgNumber.Text)
End Sub
```

Receiving a Reply

```
Private Sub POPCT1_DocOutput(DocOutput docOutput)
Dim strData as Variant
‘ retrieve the buffer
If (DocOutput.State = icDocHeaders) Then
   Dim header As DocHeader
   For Each header In DocOutput.Headers
    Debug.Print header.Name & ": " & header.Value
   Next
ElseIf (DocOutput.State = icDocData) Then
POPCT1.GetData(strData);
‘ append data as it arrives into an edit control
txtMessage = txtMessage & strData & NL
End If
End Sub
```

SMTP Client ActiveX Control Overview

The SMTP ActiveX Control is used to develop Visual Basic applications that communicate with SMTP servers to send mail. It implements the basic client SMTP Protocol as specified by RFC821, *Simple Mail Transfer Protocol*. It provides a reusable component that gives applications access to SMTP mail servers and mail posting capabilities.

The SMTP ActiveX Control supports a high-level interface that incorporates all SMTP commands used in sending out a mail message. Using this interface, a mail message can be sent with a single call.

The following is a list of properties, methods and events that are supported by the SMTP Control. For an example illustrating the use of the control in a real life situation, see SMTP Sample Session.

Property	**Method**	**Event**
Blocking	AboutBox	Busy
BlockResult	Cancel	Cancel
Busy	Connect	DocInput
DocInput	Expand	Error
EnableTimer	Help	Expand
Error	NOOP	Help
Logging	Quit	Log
NotificationMode	Reset	NOOP
ProtocolState	SendDoc	ProtocolStateChanged
ProtocolStateString	Verify	Reset
RemoteHost		StateChanged
RemotePort		TimeOut
ReplyCode		Verify
ReplyString		
SleepTime		
SocketHandle		
State		
StateString		
Timeout		
URL		

Using the SMTP Control

To use the SMTP Client ActiveX Control you must choose the SMTP toolbox icon.

There should be no speed overhead and response delay other than the one given by the network. This control uses and is dependent on the DocStream object (DocInput).

Blocking SMTP Property

Description

Indicates whether methods should block until complete or not.

Syntax

object.**Blocking** *[=Boolean]*

Permission

W (Read/Write)

Availability

D (Design) and R (Runtime)

Data Type

Boolean.

Default Value

False.

Range

True or False.

Comments

If Blocking is set to True, then a method will not be completed until the proper event(s) for a given request are fired, if applicable.

BlockResult SMTP Property

Description

Returns the result value of the last blocking method called.

Syntax

object.**BlockResult**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

BlockingResultConstants.

Default Value

icBlockOK..

Range

Name	Value	Description
icBlockOK	0	Blocking method was successful.
icTimedOut	1	Blocking method returned due to timeout.
icErrorExit	2	Blocking method returned due to an error.
icBlockCancel	3	Blocking method returned due to cancel.
icUserQuit	4	Blocking method returned due application end.

Busy SMTP Property

Description

Indicates a command is in progress.

Syntax

object.**Busy** *[= Boolean]*

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Boolean.

Default Value

N/A.

Range

N/A.

DocInput SMTP Property

Description

Object describing input information for the document being transferred.

Syntax

object.**DocInput**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

DocInput.

Default Value

N/A.

Range

N/A.

Comments

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

Properties of the DocInput object may be set before calling the SendDoc method or they may be passed as arguments to this method. The DocInput object is also used for conveying information about the progress of the document transfer and for data linking and streaming.

For more information, see DocOutput Object Events and Common Control Objects.

EnableTimer SMTP Property

Description

Enable timer for the specified event. The event is specified by entering:

```
EnableTimer(short event)
```

Syntax

object.**EnableTimer(***event***)** [= *Boolean*]

Permission

W (Write Only).

Note: This is the only control property that is Write only.

Availability

R (Runtime)

Data Type

Boolean.

Default Value

False. (The timer for this event will not be enabled.)

Range

True or False

Comments

Event is an integer value that determines the type of Timeout event that will be enabled. Constants defined for enum types for events are:

Value	Meaning
prcConnectTimeout = 1	Timeout for connect. If connection is not established within the timeout period, the Timeout event will be activated.
prcReceiveTimeout = 2	Timeout for receiving data. If no data arrives within the timeout period, the Timeout event will be activated.
prcUserTimeout= 65	Timeout for user defined event. Use prcUserTimeout + [Integer] range for custom timeout events.

Errors SMTP Property

Description

A collection of errors that can be accessed for details about the last error that occurred. This collection should be used within an Error event if information passed through the Error event is not sufficient. For more details, see icErrors Item Overview.

Syntax

object.**Errors**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

icErrors.

Default Value

N/A.

Range

N/A.

Logging SMTP Property

Description

Indicates whether log events should be fired when log data is available.

Syntax

object.**Logging** [= *Boolean*]

Permission

W (Read/Write).

Availability

D (Design) and R(Runtime).

Data Type

Boolean.

Default Value

False.

Range

True or False

NotificationMode SMTP Property

Description

Determines when notification is issued for incoming data. Notification can also be suspended.

Syntax

object.**NotificationMode** [= *Integer*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

NotificationModeConstants.

Default Value

icCompleteMode.

Range

0-maximum unsigned long. At present, the values are:

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

ProtocolState SMTP Property

Description

This property specifies the current state of the protocol.

Syntax

object.**ProtocolState**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

NNTPProtocolStateConstants.

Default Value

smtpBase.

Range

0-1. defined for the enum types of ProtocolState property are:

Value	Meaning
smtpBase = 0	Base state before connection to server is established.
smtpTrainsaction = 1	Connection to server is established. This is the valid state for calling methods on the control.

ProtocolStateString SMTP Property

Description

String representation of ProtocolState.

Syntax

object.**ProtocolStateString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

"BASE"

Range

N/A.

RemoteHost SMTP Property

Description

The remote machine to connect to if the RemoteHost parameter in the Connect method is missing. You can either provide a host name or an IP address string in dotted format. For example, “127.0.0.1”.

Syntax

object.**RemoteHost** [= *String*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

String.

Default Value

“127.0.0.1”.

Range

N/A.

RemotePort SMTP Property

Description

The remote port number to which to connect.

Syntax

object.**RemotePort** [= *Long*]

Permission

W (Read/Write).

Availability

R (Runtime) and D (Design).

Data Type

Long.

Default Value

25.

Range

1-65535.

ReplyCode SMTP Property

Description

The value of the reply code is a protocol specific number that determines the result of the last request, as returned in the ReplyString property.

Syntax

object.**ReplyCode**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

0

Range

See RFC 821 for a list of valid reply codes.

ReplyString SMTP Property

Description

Lists the last reply string sent by the SMTP Server to the client as a result of a request.

Syntax

object.**ReplyString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

SleepTime SMTP Property

Description

Specifies the sleep time between checking messages, if Blocking is True.

Syntax

object.**SleepTime** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

10 ms.

Range

>=zero.

Comments

Only applies when in Blocking mode.

SocketHandle SMTP Property

Description

Socket handle for the primary connection (Request/Reply connection).

Syntax

object.**SocketHandle**

Permission

R (Read only)

Availability

R (Runtime).

Data Type

Long.

Default Value

N/A.

Range

>=0

Comments

Some protocols require more than one connection. SocketHandle is the handle for the request/reply connection. If the SocketHandle is less than zero, it is valid.

State SMTP Property

Description

This property specifies the connection state of the control.

Syntax

object.**State**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

StateConstants.

Default Value

prcDisconnected.

Range

1-6. Constants defined for enum types of State property are:

Value	**Meaning**
prcConnecting = 1	Connecting. Connect has been requested, waiting for connect acknowledge.
prcResolvingHost = 2	Resolving Host. Occurs when RemoteHost is in name format rather than dot-delimited IP format.
prcHostResolved = 3	Resolved the host. Occurs only if ResolvingHost state has been entered previously.
prcConnected = 4	Connection established.
prcDisconnecting = 5	Connection closed. Disconnect has been initiated.
prcDisconnected = 6	Initial state when protocol object is instantiated, before Connect has been initiated, after a Connect attempt failed or after Disconnect performed.

StateString SMTP Property

Description

A string representation of State.

Syntax

object.**StateString**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

"Disconnected".

Range

N/A.

Timeout SMTP Property

Description

Timeout value for the specified event. The event is specified by entering:

```
Timeout(short event)
```

Syntax

object.**Timeout** (*event*) [= *Long*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

Long.

Default Value

0.

Range

0-maximum unsigned long. Constants defined for enum types for events are:

Value	Meaning
prcConnectTimeout = 1	Timeout for connect. If connection is not established within the timeout period, the Timeout event will be activated.
prcReceiveTimeout = 2	Timeout for receiving data. If no data arrives within the timeout period, the Timeout event will be activated.
prcUserTimeout= 65	Timeout for user defined event. Use prcUserTimeout + [Integer] range for custom timeout events.

URL SMTP Property

Description

URL string identifying the current document being transferred. The URL format for this control is:

```
SMTP://host:port
```

Syntax

object.**URL** [= S*tring*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

String.

Default Value

Empty string.

Range

Valid URL.

Comments

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.

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 smtp: string may be omitted when using the SMTP control.

AboutBox SMTP Method

Description

Shows information about this control.

Return Value

Void.

Syntax

object.**AboutBox**

Parameters

None.

Cancel SMTP Method

Description

Initiates a Cancel request to cancel a pending request. If successful, the Cancel event is called. In case of an error, the Error event is called.

Return Value

Void.

Syntax

object.**Cancel**

Parameters

None.

Comments

When completed, the state will be Disconnected.

Connect SMTP Method

Description

Initiates a Connect request. If the connection is established, StateChanged is called. In case of an error, the Error event is called.

Return Value

Void.

Syntax

object.**Connect** [*RemoteHost*], [*RemotePort*]

Parameters

RemoteHost

Optional. Remote Host to be connected.

Data Type: String

Param: IN

Default Value: N/A

RemotePort

Optional. Remote Port to be connected.

Data Type: long

Param: IN

Default Value: N/A

Comments

Optional arguments to this method will override the values from corresponding RemoteHost and RemotePort properties. The values of the properties will change. If no argument is given, the values from the properties will be used to establish the connection.

Expand SMTP Method

Description

Initiate a EXPN request. If successful, Expand event will be called when the request completes. ReplyString will contain the reply from the server. In case of an error, the Error event is called.

Return Value

Void.

Syntax

object.**Expand** [= *name*]

Parameters

name:

email id or name that will be expanded

Data Type: String

Param: IN

Default Value: N/A

Help SMTP Method

Description

Initiate a HELP request. If successful, the Help event will be called when the request completes. ReplyString will contain the reply from the server. In case of an error, the Error event is called.

Return Value

Void.

Syntax

object.**Help** [topic]

Parameters

topic

Optional. Topic you want help on.

Data Type: String

Param: IN

Default Value: N/A

NOOP SMTP Method

Description

Initiates a NOOP request. If successful, the Noop event will be called. Noop verify that the connection is alive. In case of an error, the Error event is called.

Return Value

Void.

Syntax

object.**Noop**

Parameters

None

Quit SMTP Method

Description

Initiates a Quit request to Quit the session and disconnect. In case of an error, the Error event is called.

Return Value

Void.

Syntax

object.**Quit**

Parameters

None

Reset SMTP Method

Description

Initiates a RSET request. If successful, the Reset event will be called. In case of an error, the Error event is called.

Return Value

Void.

Syntax

object.**Reset**

Parameters

None

SendDoc SMTP Method

Description

Initiates a Request to send a document identified by a URL, InputFile, or InputDate. In case of an error, the Error event is called.

Return Value

Void.

Syntax

object.**SendDoc** [*URL*], [*Headers*], [*InputData*], [*InputFile*], [*OutputFile*]

Parameters

URL

Optional. The URL identifying the remote document to be sent.

Data Type: String

Param: IN

Default Value: DocInput.URL

Headers

Optional. Headers used for sending the document. The user should directly modify the DocInput.Headers property.

Data Type: DocHeaders

Param: IN

Default Value: DocInput.Headers

InputData

Optional. A data buffer containing the document to be sent.

Data Type: VARIANT

Param: IN

Default Value: DocInput.GetData

InputFile

Optional. A local file containing the document to be sent.

Data Type: String

Param: IN

Default Value: DocInput.Filename

OutputFile

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).

Data Type: String

Param: IN

Default Value: DocOutput.Filename

Comments

The SendDoc method makes it possible to send a document. For the SMTP control this means sending a mail message to the server.

The URL and 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.

The URL type (first part up to the colon) may be omitted and will default to the correct type for this control. For example, when using the SMTP control, the "smtp" string may 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.

For more information, see Common Control Objects, DocInput Object Overview, and DocOutput Object Overview.

Verify SMTP Method

Description

Initiates a VRFY request. If successful, the Verify event will be called when the request completes. ReplyString will contain the reply from the server. In case of an error, the Error event is called.

Return Value

Void.

Syntax

object.**Verify** *name*

Parameters

name

email id or name that will be verified

Data Type: String

Param: IN

Default Value: N/A

SMTP Events

Events are used for SMTP client notification. They indicate that an action has been requested and processed. Any errors which occur during command processing result in the Error event being called with appropriate error codes. Error codes, state changes, and protocol return values are usually checked during event processing.

The following series of Help topics describe the events supported by the SMTP Client Control. Each event description includes the syntax, related parameters, their data type, default value, and whether the parameter is used for input or output (IN or OUT). For a complete list of SMTP events, see SMTP Client ActiveX Control Overview.

Busy SMTP Event

Description

This event is activated when a command is in progress or when a command has completed.

Syntax

*object*_**Busy** (*Busy* **As Boolean**)

Parameters

Busy

Indicates whether or not a command is in progress.

Data Type: Boolean. If the argument is True, a command is in progress.

Cancel SMTP Event

Description

This event is activated after a cancellation request has been completed and satisfied. After this event the object's state changes to Disconnected/Base.

Syntax

*object*_**Cancel**

Parameters

None.

DocInput SMTP Event

Description

Indicates that input data has been transferred or the DocInput state has changed.

Syntax

*object*_**DocInput** (**ByVal** *DocInput* **As DocInput**)

Parameters

DocInput

Object describing document input data for the current transfer.

Data Type: DocInput

Param: IN

Default Value: N/A

Comments

The DocInput event can be used in its basic form for notification of transfer progress, (for example, for updating a progress bar). The DocInput.BytesTotal, DocInput.BytesTransferred and DocInput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

To gain more power from this control, you can also use the DocInput event for data streaming. For more information, see DocInput Object Overview.

Error SMTP Event

Description

This event is activated when an error occurs in background processing (for example, failed to connect or failed to send or receive in the background).

Syntax

*object*_**Error** (*ErrCode* **As Integer**, *Description* **As String**, *Scode* **As Long**, *Source* **As String**, *HelpFile* **As String**, *HelpContext* **As Long**, *CancelDisplay* **As Boolean**)

Parameters

ErrCode

The short error code. For a list of possible SMTP error codes see SMTP Error Codes.

Description

String containing error information.

sCode

The long Scode.

Source

Error source.

HelpFile

Help file name.

HelpContext

Help file context.

CancelDisplay

Indicates whether to cancel the display. The default is TRUE (no display of the default error message box). If you do want to use the default message box, set CancelDisplay to FALSE.

Expand SMTP Event

Description

This event is activated after the successful completion of a Expand request.

Syntax

*object*_**Expand**

Parameters

None.

Help SMTP Event

Description

This event is activated after the successful completion of a Help request.

Syntax

*object*_**Help**

Parameters

None.

Log SMTP Event

Description

This event is activated when logging data is available.

Syntax

*object*_**Log**

Parameters

None.

NOOP SMTP Event

Description

This event is activated after the successful completion of a Noop request.

Syntax

*object*_**Noop**

Parameters

None.

ProtocolStateChanged SMTP Event

Description

This event is activated whenever the protocol state changes.

Syntax

*object*_**ProtocolStateChanged** (*ProtocolState* **As Integer**)

Parameters

Refer to the ProtocolState property and ProtocolStateString for possible values of the ProtocolState parameter.

Reset SMTP Event

Description

This event is activated after the successful completion of a Reset request.

Syntax

*object*_**Reset**

Parameters

None.

StateChanged SMTP Event

Description

This event is activated whenever the state of the transport state changes.

Syntax

*object*_**StateChanged** (*State* **As Integer**)

Parameters

Refer to the State property and StateString for possible values of the state parameter.

TimeOut SMTP Event

Description

This event is activated when the timer for the specified event expires.

Syntax

*object*_**TimeOut** (ByVal *Event* **As Integer,** *Continue* **As Boolean**)

Parameters

Event

Defines the event to which the time interval applies.

Data Type: Short

Continue

Determines if the timer is active or not. Set *Continue* to TRUE to keep the timer active.

Data Type: Boolean

Default Value: False

Comments

Event is an integer value that determines the type of Timeout event that will be enabled. Constants defined for enum types for events are:

Value	**Meaning**
prcConnectTimeout = 1	Timeout for connect. If connection is not established within the timeout period, the Timeout event will be activated.
prcReceiveTimeout = 2	Timeout for receiving data. If no data arrives within the timeout period, the Timeout event will be activated.
prcUserTimeout= 65	Timeout for user defined event. Use prcUserTimeout + [Integer] range for custom timeout events.

For more information, see Timeout SMTP Property.

Verify SMTP Event

Description

This event is activated after the successful completion of a Verify request.

Syntax

*object*_**Verify**

Parameters

None.

SMTP Sample Session

The SMTP Client Control is used to send a mail message to a server. A code fragment from a simple application using the SMTP control might look like this:

```
DocInput.headers.Add “From”, “joe@xyz.com”
DocInput.headers.Add “To”, “adam@abc.com”
DocInput.headers.Add “CC”, “sue@acme.com”
DocInput.headers.Add “Subject”, “project scheduling”
Dim mailText as String

mailText = “Please give me your time estimates to be included into our project
schedule.”

Smtpct1.RemoteHost = “mail”

Smtpct1.SendDoc,,mailText
```

WinSock ActiveX Controls

There are two WinSock controls supported by ActiveX: TCP and UDP. Click on one of topics below for additional information.

- WinSock TCP ActiveX Control Overview
- \WinSock UDP ActiveX Control Overview

WinSock TCP ActiveX Control Overview

The WinSock TCP ActiveX Control implements the Transmission Control Protocol for both client and server applications. Invisible to the user, the WinSock TCP control provides easy access to TCP network services. It can be used by Visual Basic, Delphi, and C++ programmers.

To write client and/or server applications, you do not need to understand the details of TCP or to call low-level WinSock APIs. By setting properties and calling methods on the control, you can easily connect to a remote machine and exchange data in both directions. Events are used to notify you of network activities.

The following table lists the properties, methods, and events used by the TCP ActiveX Control. For an example illustrating the use of the control in a real life situation, see TCP Sample Session.

Property	Method	Event
BytesReceived	AboutBox	Close
LocalHostName	Accept	Connect
LocalIP	Close	ConnectionRequest
LocalPort	Connect	DataArrival
RemoteHost	GetData	Error
RemoteHostIP	Listen	SendComplete
RemotePort	PeekData	SendProgress
SocketHandle	SendData	
State		

Using the TCP Control

To use the TCP ActiveX Control you must choose the TCP toolbox icon. You must also use the correct syntax.

BytesReceived TCP Property

Description

It shows the amount of data received (currently in the receive buffer). The GetData method should be used to retrieve data.

Syntax

object.**BytesReceived**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

0.

Range

0 - 0xFFFFFFFF

LocalHostName TCP Property

Description

Local machine name.

Syntax

object.**LocalHostName**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

LocalIP TCP Property

Description

The IP address of the local machine. It has the format:

```
number.number.number.number
```

Syntax

*object.***LocalIP**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

LocalPort TCP Property

Description

For the client, this designates the local port to use. Specify port 0 if the application does not need a specific port. In this case, the control will select a random port. After a connection is established, this is the local port used for the TCP connection.

For the server, this is the local port to listen on. If port 0 is specified, a random port is used. After calling the Listen method, the property contains the actual port that has been selected.

Syntax

object.**LocalPort** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

0.

Range

0 - 65535

RemoteHost TCP Property

Description

The remote machine to connect to if the RemoteHost parameter of the Connect method is not specified. You can either provide a host name or an IP address string in dotted format.

Syntax

object.**RemoteHost** [= *String*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

RemoteHostIP TCP Property

Description

For the client, after a connection has been established (i.e., after the Connect event has been activated), this property contains the IP string of the remote machine in dotted format.

For server, after an incoming connection request (ConnectionRequest event), this property contains the IP string (in dotted format) of the remote machine initiating the connection.

Syntax

*object.***RemoteHostIP**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

RemotePort TCP Property

Description

For the client, this is the remote port number to which to connect if the RemotePort parameter of the Connect method is not specified.

For the server, after an incoming connection request event (ConnectionRequest has been activated), this property contains the port that the remote machine uses to connect to this server.

Syntax

*object.***RemotePort** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

0.

Range

0 - 65535

SocketHandle TCP Property

Description

This is the socket handle the control uses to communicate with the WinSock layer.

Syntax

*object.***SocketHandle**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Long.

Default Value

-1.

Range

-1 - 65535

Comment

This property is for advanced programmers. You can use SocketHandle in direct WinSock API calls. However, you should be aware that if WinSock calls are used directly, certain events may not be activated appropriately. A value of less than zero indicates that the Sockethandle is not available.

State TCP Property

Description

The state of the control, expressed as an enum type.

Syntax

*object.***State**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Integer.

Default Value

0.

Range

0-9. Constants defined for enum types in this property are:

Enum Type	**Meaning**
sckClosed = 0	Closed
sckOpen = 1	Open
sckListening = 2	Listening
sckConnectionPending = 3	Connection pending
sckResolvingHost = 4	Resolving host
sckHostResolved = 5	Host resolved
sckConnecting = 6	Connecting
sckConnected = 7	Connected
sckClosing = 8	Peer is closing the connection
sckError = 9	Error

TCP Property Page

One property page is provided for viewing and editing the following properties:

- RemoteHost
- RemotePort
- LocalPort

AboutBox TCP Method

Description

Shows information about this control.

Return Value

Void.

Syntax

object.**AboutBox**

Parameters

None.

Accept TCP Method

Description

This method is used to accept an incoming connection when handling a ConnectionRequest event.

Return Value

Void.

Syntax

object.**Accept** *RequestID*

Parameters

RequestID

The incoming connection request identifier. This should be the requestID passed in the ConnectionRequest event.

Data Type: Long

Param: IN

Comment

Accept should be used on a new control instance (other than the one that is in the listening state.)

See Also

ConnectionRequest TCP Event

Close TCP Method

Description

Closes a TCP connection or a listening socket for both client and server.

Return Value

Void.

Syntax

*object.***Close**

Parameters

None.

Connect TCP Method

Description

Initiates connection to remote machine.

Return Value

Void.

Syntax

*object.***Connect** [*RemoteHost*], [*RemotePort*]

Parameters

RemoteHost

Optional. If this parameter is missing, Connect will connect to the remote host specified in the RemoteHost property.

Data Type: VARIANT

Param: IN

RemotePort

Optional. If this parameter is missing, Connect will connect to the remote port specified in the RemotePort property.

Data Type: VARIANT

Param: IN

Comments

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

GetData TCP Method

Description

Retrieves data.

Return Value

Void.

Syntax

object.**GetData** *data* [*type*], [*maxLen*]

Parameters

Data

Stores retrieved data after the method returns successfully. If there is not enough data available for requested type, *data* will be set to Empty.

Data Type: VARIANT

Param: OUT

Type

Optional. Type of data to be retrieved.

Data Type: VARIANT

Param: IN

Default Value: vbArray + vbByte

Currently, the following variant types are supported.

Type	C++	VB 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

maxLen

Optional length parameter. This parameter can be used to specify the desired size when receiving a byte array or a string . If this parameter is missing for byte array or string, all available data will be retrieved. If provided, for data types other than byte array and string, this parameter is ignored.

Data Type: VARIANT

Param: IN

Default Value: All data available.

Comments

If the type is specified as vbString, string data is converted to UNICODE before returning to the user.

Listen TCP Method

Description

It includes creating a socket and putting the socket in the listening mode.

Return Value

Void.

Syntax

*object.***Listen**

Parameters

None.

Comment

When there is an incoming connection, the ConnectionRequest event is activated. When handling ConnectionRequest, the application should use the Accept method (on a new control instance) to accept the connection.

PeekData TCP Method

Description

Similar to GetData except PeekData does not remove data from input queue.

Return Value

Void.

Syntax

object.**PeekData** *data*, [*type*], [*maxLen*]

Parameters

Data

Stores retrieved data after the method returns successfully. If there is not enough data available for requested type, *data* will be set to Empty.

Data Type: VARIANT

Param: OUT

Type

Optional. Type of data to be retrieved.

Data Type: VARIANT

Param: IN

Default Value: vbArray + vbByte

Currently, the following variant types are supported.

Type	**C++**	**VB 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

maxLen

Optional length parameter. This parameter can be used to specify the desired size when receiving a byte array or a string . If this parameter is missing for byte array or string, all available data will be retrieved. If provided, for data types other than byte array and string, this parameter is ignored.

Data Type: VARIANT

Param: IN

Default Value: All data available.

Comments

If the type is specified as vbString, string data is converted to UNICODE before returning to the user.

SendData TCP Method

Description

Sends data to peer.

Return Value

Void.

Syntax

*object.***SendData** *data*

Parameters

Data

Data to be sent. For binary data, byte array should be used.

Data Type: VARIANT

Param: IN

Currently, the following variant types are supported.

Type	**C++**	**VB 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

Comments

When a UNICODE string is passed in, it is converted to an ANSI string before being sent out on the network.

The user is notified of sending progress by the SendProgress and SendComplete event.

Close TCP Event

Description

The event is activated when the peer closes the connection. Applications should use the Close method to correctly close the connection.

Syntax

*object*_**Close**

Parameters

None.

Connect TCP Event

Description

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

Syntax

*object*_**Connect**

Parameters

None.

ConnectionRequest TCP Event

Description

The event is activated when there is an incoming connection request. RemoteHostIP and RemotePort properties store the information about the client after the event is activated.

The server can decide whether or not to accept the connection. If the incoming connection is not accepted, the peer (client) will get the Close event. Use the Accept method (on a new control instance) to accept an incoming connection.

Syntax

*object*_**ConnectionRequest** (*RequestID* **As Long***)*

Parameters

RequestID

The incoming connection request identifier. This parameter should be passed to the Accept method on the second control instance.

Data Type: Long

Param: IN

DataArrival TCP Event

Description

The event is activated when new data arrives.

Syntax

*object*_**DataArrival** (*BytesTotal* **As Long**)

Parameters

BytesTotal

The total amount of data that can be retrieved.

Data Type: Long

Param: IN

Comments

This event will not be activated if you do not retrieve all the data in one GetData call. It is activated only when there is new data. You can always use the BytesReceived property to check how much data is available at any time.

Error TCP Event

Description

This standard error event is activated whenever an error occurs in background processing (for example, failed to connect, or failed to send or receive in the background).

Syntax

*object*_**Error** (*ErrCode* **As Integer**, *Description* **As String**, *Scode* **As Long**, *Source* **As String**, *HelpFile* **As String**, *HelpContext* **As Long**, *CancelDisplay* **As Boolean**)

Parameters

ErrCode

An integer that defines the error code. For a list of possible WinSock error codes see WinSock Error Codes.

Description

String containing error information.

Scode

The long SCODE.

Source

String describing the error source.

HelpFile

String containing help file name.

HelpContext

Help file context.

CancelDisplay

Indicates whether to cancel the display. The default is TRUE (no display of the default error message box). If you do want to use the default message box, set CancelDisplay to FALSE.

SendComplete TCP Event

Description

The event is activated when the send buffer is empty.

Syntax

*object*_**SendComplete**

Parameters

None.

SendProgress TCP Event

Description

This event notifies the user of sending progress. It is activated when more data has been accepted by the stack.

Syntax

*object*_**SendProgress** (*BytesSent* **As Long**, *BytesRemain* **As Long**)

Parameters

BytesSent

The number of bytes that have been sent since the last time this event was activated.

Data Type: Long

Param: IN

BytesRemain

The number of bytes in the send buffer waiting to be sent.

Data Type: Long

Param: IN

TCP Localization

The resources for the control's about box, property page, and strings are in resource DLL nmorenu.dll. The resource DLL is localized for each language.

TCP Sample Session

The following session samples illustrates a real-life scenario using the TCP Control. From this example you can see how to write for both the client and server.

{button ,JI(`NIA.HLP',`IDH_TCP_Client')} **TCP Client**

{button ,JI(`NIA.HLP',`IDH_TCP_Server')} **TCP Server**

TCP Client

A TCP client actively initiates a connection to a remote machine. Set the RemoteHost and RemotePort property. Then, call the Connect method, as shown here.

```
TCP1.RemoteHost = “john@somecompany.com”
TCP1.RemotePort = 7         ‘ the echo port
TCP1.Connect
```

When the connection has been established successfully, a Connect event occurs. If the remote host rejected the connection, a Error event occurs.

```
Private Sub TCP1_Connect()
  StatusBar1.status = “Connected”
End Sub

Private Sub TCP1_Error(Number As Integer, Description As String, Scode As Long,
Source As String, HelpFile As String, HelpContext As Long, CancelDisplay As
Boolean)
  StatusBar1.status = Number & “ - “ & Description
  CancelDisplay = True
End Sub
```

After a connection has been established, use the SendData method to stream data to a remote machine. A DataArrival event occurs when there is incoming data. Use the Close method to terminate the connection, as illustrated in the following code:

```
Private Sub btnSend_Click()
  TCP1.Send “My name is Mary.”
End Sub


Private Sub TCP1_DataArrival(ByVal bytesTotal As Long)
Dim data As Variant
  TCP1.GetData data,vbString       ‘ retrieve data
  txtRecv = data                   ‘ update display
  TCP1.Close               ‘ close the connection
End Sub
```

TCP Server

A TCP server listens at a particular port for incoming connections. To write an echo server which echoes back all data it receives, the server would listen at the standard echo port 7. You should set LocalPort to 7 and call the Listen method. When there is an incoming connection request, a ConnectionRequest event occurs. Use another instance of TCP control to accept the incoming connection. When this is complete, the application can send and receive data on the newly accepted connection as described in TCP Client section. The following example illustrates how to make the server connection.

```
TCPSvr.LocalPort = 7
TCPSvr.Listen
StatusBar1.status = "Listening"


Private Sub TCPSvr_ConnectionRequest(ByVal requestID As Long)
  TCPAccepted.Accept (requestID)          ‘ TCPAccepted is a new control instance
  ' we are only accepting one connection in this sample. let’s close the
listening control
  TCPSvr.Close
  StatusBar1.status = "Incoming connection accepted"
End Sub


Private Sub TCPAccepted_DataArrival(ByVal bytesTotal As Long)
Dim data As Variant
  TCPAccepted.GetData data          ‘ retrieve data
  TCPAccepted.SendData data         ‘ echo back
End Sub


Private Sub TCPAccepted_Close()
  TCPAccepted.Close
End Sub
```

WinSock UDP ActiveX Control Overview

The WinSock UDP (User Datagram Protocol) ActiveX Control, invisible to the user, provides easy access to UDP network services. It implements WinSock for both client and server and represents a communication point utilizing UDP network services. It can also be used to send and retrieve UDP data.

The WinSock UDP control can be used by Visual Basic, Delphi, and C++ programmers. To write UDP applications, you do not need to understand the details of UDP or to call low-level WinSock APIs. By setting properties and calling methods on the control, you can easily send data to a remote machine or retrieve data from the network. Events are used to notify users of network activities.

The following table lists the properties, methods, and events supported by the UDP Control. For an example illustrating the use of the control in a real life situation, see UDP Sample Session.

Property	Method	Event
LocalHostName	AboutBox	Error
LocalIP	GetData	DataArrival
LocalPort	SendData	
RemoteHost		
RemoteHostIP		
RemotePort		
SocketHandle		

Using the UDP Control

To use the UDP ActiveX Control you must have the UDP toolbox icon.

LocalHostName UDP Property

Description

This property defines the local machine name.

Syntax

*object.***LocalHostName**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

LocalIP UDP Property

Description

This property specifies the IP address of the local machine. It has the format:

```
”number.number.number.number”
```

Syntax

object.**LocalIP**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

LocalPort UDP Property

Description

Designates the local port to use.

Syntax

object.**LocalPort** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

0.

Range

0 - 65535

RemoteHost UDP Property

Description

The remote machine to which to send UDP data. You can enter either a host name or an IP address string in dotted format (for example, “127.0.0.1”).

Syntax

object.**RemoteHost** [= *String*]

Permission

W (Read/Write).

Availability

D (Design)   and R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

RemoteHostIP UDP Property

Description

After the GetData method, this property contains the IP address of the remote machine sending the UDP data.

Syntax

object.**RemoteHostIP**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A.

RemotePort   UDP Property

Description

This property specifies the remote port number on the remote machine to which UDP data is sent. After the GetData method, this property contains the remote port that is sending the UDP data.

Syntax

object.**RemotePort** [= *Long*]

Permission

W (Read/Write).

Availability

D (Design) and R (Runtime).

Data Type

Long.

Default Value

0.

Range

0 - 65535

SocketHandle UDP Property

Description

This is the socket handle the control uses to communicate with the WinSock layer.

Syntax

object.**SocketHandle**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Long.

Default Value

-1.

Range

-1 - 65535

Comments

This property is for advanced programmers. You can use SocketHandle in direct WinSock API calls. However, you should be aware that if WinSock calls are used directly, certain events may not be activated appropriately. A value less than zero indicates that the Sockethandle is not available.

UDP Property Page

One property page is provided for viewing and editing the following properties:

- RemoteHost
- RemotePort
- LocalPort

AboutBox UDP Method

Description

Shows information about this control.

Return Value

Void.

Syntax

object.**AboutBox**

Parameters

None.

GetData UDP Method

Description

Retrieves data.

Return Value

Void.

Syntax

object.**GetData** *data,* [*type*]

Parameters

Data

Stores retrieved data after the method returns successfully. If there is no data available, *data* will be set to Empty.

Data Type: VARIANT

Param: OUT

Type

Type of data to be retrieved. It can be either vbString or byte array.

Data Type: VARIANT

Param: IN

Default Value: vbArray + vbByte

Comments

If the type is specified as vbString, string data is converted to UNICODE before returning to the user.

SendData UDP Method

Description

This method sends data to remote machine.

Return Value

Void.

Syntax

object.**SendData** *data*

Parameters

Data

Data to be sent. For binary data, byte array should be used.

Data Type: VARIANT

Param: IN

Currently, the following variant types are supported.

Type	C++	VB 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

Comments

The RemoteHost and RemotePort properties should be set before calling this method. When a UNICODE string is passed in, it is converted to an ANSI string before being sent out on the network.

DataArrival UDP Event

Description

The event is activated when a new data packet arrives.

Syntax

*object*_**DataArrival** (*BytesTotal* **As Long**)

Parameters

BytesTotal

The total amount of data, in bytes, currently available.

Data Type: Long

Param: IN

Error UDP Event

Description

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

Syntax

*object_***Error** (*ErrCode* **As Integer**, *Description* **As String**, *Scode* **As Long**, *Source* **As String**, *HelpFile* **As String**, *HelpContext* **As Long**, *CancelDisplay* **As Boolean**)

Parameters

ErrCode

An integer that defines the error code. For a list of possible WinSock error codes see WinSock Error Codes.

Description

String containing error information.

Scode

The long SCODE.

Source

String describing the error source.

HelpFile

String containing help file name.

HelpContext

Help file context.

CancelDisplay

Indicates whether to cancel the display. The default is TRUE (no display of the default error message box). If you do want to use the default message box, set CancelDisplay to FALSE.

UDP Localization

The resources for the control's about box, property page, and strings are in resource DLL nmorenu.dll. The resource DLL is localized for each language.

UDP Sample Session

The following session illustrates a real life scenario using the UDP Control.

To specify the local UDP port to use, you, the Visual Basic developer would set the LocalPort property.

```
UDP1.LocalPort= 9           ‘ discard port
```

To send UDP data to a remote machine, you need to set the RemoteHost and RemotePort properties, and call the SendData method as shown here:

```
UDP1.RemoteHost = “john@somecompany.com”
UDP1.RemotePort = 7         ‘ the echo port
UDP1.SendData “Hello!”
```

When data arrives at the local port the control is using, the DataArrival event is activated. GetData can be used to retrieve data as in the following code.

```
Private Sub UDP1_DataArrival(ByVal bytesTotal As Long)
Dim data As Variant
  UDP1.GetData data               ‘ retrieve data
  txtRecv = data                  ‘ update display
End Sub
```

Introduction to NEWT Intranet ActiveX

Welcome to the online Software Developer's Guide for the NetManage NEWT Intranet ActiveX. This online guide is designed to help independent software vendors (ISVs) and original equipment manufacturers (OEMs) develop sophisticated and extensible applications on multiple platforms and conform to the programming model of the Windows operating system.

This guide is part of a development kit that includes the OLE interface specifications for the following ActiveX Controls:

- Basic WinSock Controls
- Client Controls
- Server Controls
- WEB Controls.
- Infrastructure Controls and COM objects

The user is assumed to understand and be familiar with Dynamic Link Library (DLL) objects, Visual Basic and C/C++ programming language concepts, Windows application development, and TCP/IP protocol terminology.

How to Use This Help System

To obtain maximum benefit from the NetManage NEWT Intranet ActiveX Software Developer's Guide, first read the topics in this section (accessible using the browse buttons), then use the Index or Find features in the Windows Help window to find the specific information you need. These introductory topics describe the ActiveX control groups, making it easier for you to categorize and work with the ActiveX technology.

The NEWT Intranet ActiveX Software Developer's Guide describes the ActiveX controls, providing an introduction to each control, an actual scenario describing how a Visual Basic user might use the control, and a description of each property, method and event.

For most examples described in this guide, the syntax is provided in Visual Basic language format. In your source files, the function name must be spelled exactly as shown in the syntax line, and parameters must be entered in the order shown.

Conventions

The following conventions are used in this Help system:

Convention	Meaning
Bold text	Indicates a term or character to be typed literally, such as a resource-definition statement or function (for example, **FILE** or **FTPClose**). You must type these terms exactly as shown.
Italic text	Indicates a placeholder or variable. You must provide the actual value (for example, the statement SetCursorPos(*X*,*Y*) requires that you add actual values for the X and Y parameters.
[ ]	Encloses optional parameters.
|	Separates an either or choice.
...	Represents omitted text before or after existing text.
`Monospace font`	Identifies code examples and shows syntax spacing.

Readme File

The Readme file contains installation instructions, directory structures, product change information and documentation changes that have occurred since the last version of the Software Developer's Guide. We recommend that you read the Readme file before installing any of the controls.

Installation Requirements

This section lists the hardware and software requirements for installing NEWT Intranet ActiveX.

Before you proceed, make sure you have the following:

- Windows NT or Windows '95 installed
- IBM 386/486 Based, PS/2 or compatible (minimum).

You do not need any additional tools to run the samples provided in executable form (.exe files).

Customer Support

NetManage provides technical support through e-mail and by telephone. For telephone support, you can reach us at these locations or call your local NetManage representative:

USA

Location	Phone	Fax
Cupertino, California	(408) 973-8181	(408) 973-8272
Nashua, New Hampshire	(603) 888-3500	(603) 888-0304

International

Location	Phone	Fax
Haifa, Israel	+972-4-8550-234	+972-4-8550-122
Paris, France	+33-1-47 72 08 08	+33-1-42 04 65 99
Munich, Germany	+49-(0) 816-594-700	+49-(0) 816-594-70147
London, U.K.	+44-483-302333	+44-483-302999
Tokyo, Japan	+81- (0)3-3221-8400	+81- (0)3-3221-8484

E-Mail

You can also receive support by e-mail. Within the United States please refer any question regarding the SDK to: dev_support@netmanage.com. All other areas should contact international support at: intl_support@netmanage.com.

Your Comments Are Welcome

We value your comments as a user of our NEWT Intranet ActiveX Controls. As we write and revise our online Help, your suggestions are a valuable source of information for us. Please tell us what was helpful and what needs to be improved in the online Help system. Send your comments email to us via e-mail at doc@netmanage.com.

About NEWT Intranet ActiveX

NetManage NEWT Intranet ActiveX is a set of components and mechanisms based on ActiveX (previously known as OLE Custom Control) technology that makes programming for the Internet both easy and powerful.

All users can use ActiveX to enhance their applications with Intranet features and make them Intranet-enabled. Intranet ActiveX controls can also be used by an expert programmer to develop complete applications for the Internet and to use all features of Internet protocols.

NEWT Intranet ActiveX can be used to build applications such as customer support systems, sales management, human resources, and other information sharing applications. It lets corporate developers build customized business solutions for use within their corporate internal network to run on top of Windows NT and Windows '95. NEWT Intranet ActiveX works with Visual Basic 4.0, Delphi, and any other development environment that supports ActiveX Controls.

ActiveX Control Groups

Controls shipped in this release consist of the following groups:

Basic WinSock Controls

TCP/UDP support

Client Controls

These controls provide support for miscellaneous Internet protocols at the client side, such as POP3 and NNTP.

Server Controls

Controls that support server-side Internet protocols.

Web Controls

These controls provide support for the World Wide Web.

Infrastructure Controls and COM Objects

Components used to support all the other controls, the sharing of data, document streaming using DocStream objects, and MIME header parsing used in many Internet protocols.

Summary of ActiveX Controls

Control Category	Description
Basic WinSock Controls	TCP/UDP Control
Client Controls	FTP, Internet, NNTP, HTTP, POP3, SMTP
WEB Controls	CGI, HTML
Infrastructure Controls	DocStream, MIME
Server	HTTP, Internet

Supported Environments

The NEWT Intranet ActiveX has been tested with the following development and run-time environments, running on both Windows 95 and Windows NT 3.51. It is compatible with all other environments that support ActiveX Control components, such as:

- Visual Basic 4.0
- MSVC 4.0
- Microsoft Access
- Microsoft Excel
- Delphi 2.01

ActiveX Control Architecture

The following are some of the elements that comprise the ActiveX architecture.

- DocStream
- DocInput and DocOutput Objects
- icError Object and Collection
- DocHeader Object and Collection

DocStream

NEWT Intranet ActiveX is designed as a family of controls with a few common interfaces that allow you to develop applications quickly. Familiarizing yourself with these common interfaces can save you time later.

See Also

DocInput and DocOutput Objects
icError Object and Collection
DocHeader Object and Collection

DocInput and DocOutput Objects

An innovative feature of NEWT Intranet ActiveX is the DocInput object and the DocOutput object. These objects have properties and methods that allow you to stream data from one control to another. For example, you can use the DocOutput object to automatically stream data from an HTTP server to an FTP control when you wish to archive the data on an FTP server. Most controls have access to either DocInput, DocOutput, or both objects.

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. Since errors may be numerous, the icError object is stored in a collection for later retrieval.

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
```

Each line in the above example would be contained in a single DocHeader object, and all lines would be part of a DocHeaders collection.

Properties, Methods, and Events

All controls are based on clearly defined interfaces consisting of properties, methods, and events.

- **Properties** indicate specific attributes such as remote host and port number. They may be set in a properties form before running each control.
- **Methods** perform an action on the object, such as connect or authenticate.
- **Events** are notifications generated after a particular action has been performed successfully. For example, each control has a StateChanged event to notify you when the current state has changed. A common scenario when working with ActiveX controls is to set properties, run methods and wait for an event to occur (fire).

Action	Example
Set Properties	Myclient.RemoteHost = "mail"
	Myclient.UserName = "josmith"
Method	Connect
	Authenticate
Event	ProcessData

Note: Read-only properties are only available at run-time, and are not listed in the Visual Basic Properties window for the control. However, they can be viewed using the Object Browser. You should also note that some common ActiveX properties of the control, such as Name, Index, About Box, and others, may appear in the Object Browser but are not documented here.

Object Descriptions

Each object description includes an explanation of the properties, methods, and events used by the object. Please click on one of the following for more information:

- Property Descriptions
- Method Descriptions
- Event Descriptions

Property Descriptions

All property descriptions may contain any of the following, wherever applicable:

Description	Describes the property or event function.
Syntax	Shows the structure of each property.
Permission	Defines access rights. R=Read only, W=read/write
Availability	Shows when property is visible. R=Runtime; D=Design. Design indicates that a property is accessible during both run time and design time.
Data Type	Defines the type of data the control expects to receive.
Default Value	Defines value used by the control if no value is entered.
Range	Defines the legal values that may be entered.
Comments	Provides additional information.

See Also
Method Descriptions
Event Descriptions

Method Descriptions

Method descriptions contain the following information.

Description	Describes the method.
Return Value	Describes the type of value returned by the method.
Syntax	Shows the structure of each method.
Parameter	Defines the parameters that should be entered. Each parameter description contains the data type, default value, and whether the parameter is used for input or output (IN or OUT).
Comments	Provides additional information, when applicable.

See Also

Property Descriptions
Event Descriptions

Event Descriptions

Event descriptions include the syntax, related parameters, data type, default value, and whether the parameter is used for input or output (IN or OUT).

See Also
Property Descriptions
Method Descriptions

Common Control Objects

A number of objects are common to all NEWT Intranet ActiveX Controls. These objects form a basis for the design of the client protocols. Having a family of controls with a few common interfaces enables you to develop applications quickly. Familiarizing yourself with these common interfaces can save you time later.

Common Control Objects include:

- DocInput
- DocOutput
- DocHeaders Collection
- icErrors Collection
- icError Item Overview

Note: A number of objects, properties, and collection names are preceded by ic, which is an abbreviation for Internet Components. This is to avoid confusion with other commonly used names.

DocInput and DocOutput describe two major categories of DocStream architecture that provides a shell for creating your own document stream for data transfer. Each of these has its own set of properties and methods.

DocInput Object Overview

The DocInput object describes input information for a document being transferred. The DocInput property is part of all controls with document input capabilities. In such controls, it is also an argument of the DocInput event. The DocInput event is documented here even though it is an event of the particular control, not of the DocInput object itself.

The following table summarizes the properties, methods and events supported by the DocInput Object. For an explanation of the description categories, see Object Descriptions.

Property	Method	Event
BytesTotal	GetData	DocInput
BytesTransferred	PushStream	
DocLink	SetData	
FileName	Suspend	
Headers		
PushStreamMode		
State		
Suspended		

BytesTotal DocInput Object Property

Description

Total bytes to be transferred or zero, if not available.

Syntax

*object.***DocInput.BytesTotal**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Long.

Default Value

Zero.

Range

>= zero

Comments

The BytesTotal property is available as soon as document transfer begins. This value may be zero if the size of the document is unknown. It may be set just prior to the icDocEnd state once the final bytes transferred is known.

BytesTransferred DocInput Object Property

Description

Number of bytes transferred so far.

Syntax

*object.***DocInput.BytesTransferred**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Long.

Default Value

Zero.

Range

>= zero

Comments

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 event is activated. 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).

DocLink DocInput Object Property

Description

Copy of another object's DocOutput DocLink property when data linking is used or empty if data linking is not used.

Syntax

*object.***DocInput.DocLink** [= *object2.DocOutput.DocLink*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

DocLink.

Default Value

Empty.

Range

Empty or a reference to DocOutput.DocLink property

Comments

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

Input data for SendDoc may be supplied only 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.

FileName DocInput Object Property

Description

Name of a local file containing the document to be transferred.

Syntax

*object.***DocInput.FileName** [**=** *String*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

Valid file name.

Comments

The FileName property may be set before calling the SendDoc method in a particular control or it may be passed as an argument to this method. If it is passed as an argument, the DocInput.FileName property will be set to the argument value.

Input data for SendDoc may be supplied only 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.

Headers DocInput Object Property

Description

A collection of headers describing the current document being transferred.

Syntax

*object.***DocInput.Headers**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

DocHeaders.

Default Value

N/A.

Range

N/A.

Comments

The contents of the Headers collection may be modified before calling the SendDoc method of the specific control.

The Headers collection contains DocHeader items, each of which represents a MIME 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.

PushStreamMode DocInput Object Property

Description

Indicates whether the stream is in push or pull mode. This property may be set by anyone implementing data streaming.

Syntax

*object.***DocInput.PushStreamMode** [= *Boolean*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

Boolean.

Default Value

False.

Range

True or False.

Comments

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, which is the pull mode) and data is specified by setting the FileName property.
- Set the PushStreamMode property to True before initiating the document transfer. See the PushStream method for more information on this technique.

PushStreamMode should only be set to TRUE once prior to beginning a PushStream sequence. Each time PushStreamMode is set TRUE, the state of the DocStream (either DocInput or DocOutput) is reset to icDocNone to ensure proper state sequencing. Setting PushStreamMode to FALSE does not affect the state of the DocStream.

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

State DocInput Object Property

Description

Indicates current state of the document transfer.

Syntax

*object.***DocInput.State**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

DocStateConstants

Default Value

icDocNone (0).

Range

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)

Suspended DocInput Object Property

Description

Indicates whether document transfer is currently suspended or not.

Syntax

*object.***DocInput.Suspended**

Permission

R (Read Only).

Availability

R (Runtime).

Data Type

Boolean.

Default Value

False.

Range

True or False

Comments

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

GetData DocInput Object Method

Description

Retrieve the current block of data to be transferred when the DocInput event is activated.

Return Value

Void.

Syntax

*object.***DocInput.GetData** *Data* [,*Type*]

Parameters

Data

Stores retrieved data after the method returns successfully. If there is no data available for the requested type, *data* is set to Empty.

Data Type: VARIANT

Param: OUT

Type

Optional. Type of data to be retrieved. The variant types that are supported are the same as listed in the DocInput.SetData method

Data Type: VARIANT

Param: IN

Default Value: vbString

Comments

The GetData method should be called only during handling of the DocInput event, when the State property is set to icDocData (3). GetData may be called to examine data during transfer when using an input file (FileName property) or input link (DocLink property).

PushStream DocInput Object Method

Description

Performs the next step of the document transfer. This method may be called by the user who implements data streaming.

Return Value

Void.

Syntax

*object.***DocInput.PushStream**

Parameters

None.

Comments

User implementation of data streaming is performed by using the PushStreamMode property and the PushStream method. These interfaces are only important to users who implement data streaming.

Input data streaming may be implemented in two ways:

- The PushStreamMode property is set to False (the default), and data is available when the DocInput event is activated. You should not call PushStream. PushStreamMode is false when FileName is used.
- PushStreamMode is set to True, and when data is available you can use Setdata to set the data and then 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, you can set document information before calling PushStream. For example, when transferring data, the SetData method may be called before calling PushStream. If the DocInput state is currently icDocData, as long as there is data to be transferred (e.g., SetData is set to a non-empty value), DocInput will remain in the icDocData state. Once SetData is used to set the data to an empty string and PushStream is called, then the state is changed to icDocEnd.

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

SetData DocInput Object Method

Description

Specify the next data buffer to be transferred when the DocInput event is activated.

Return Value

Void.

Syntax

*object.***DocInput.SetData** *Data*

Parameters

Data

Next block of data to be sent. For binary data, byte array should be used.

Data Type: VARIANT

Param: IN

Currently, the following variant types are supported.

Type	C++	VB 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

Comments

SetData may be called during DocInput event handling (when the State property is set to icDocData) to specify the next buffer of data to be transferred. SetData may also be called before calling SendDoc to specify the initial buffer of data to be transferred. The second method is an alternative to passing the InputData parameter to SendDoc. If you implement data streaming using PushStreamMode, you can also call SetData before calling PushStream.

When using an input file (FileName property) or input link (DocLink property), SetData may be called during DocInput event handling to change the next buffer of data to be transferred, if desired. Calling SetData in these cases will modify the data transferred to the target document.

Suspend DocInput Object Method

Description

Suspends or resumes document transfer.

Return Value

Void.

Syntax

*object.***DocInput.Suspend** *Suspend*

Parameters

Suspend

Indicates whether to suspend or resume transfer. If True, transfer is suspended. If False, transfer is resumed.

Data Type: Boolean

Param: IN

Comments

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.

DocInput Object Events

The DocInput object is not a control, therefore, it has no events. However, it is almost always associated with a control that has a DocInput event. The DocInput object is always a parameter of the DocInput event, and the two may be used together to perform data streaming as well as to monitor progress of the document transfer.

DocInput DocInput Object Event

Description

Indicates that the DocInput state has either changed or a data block is ready to be transferred.

Syntax

*object*_**DocInput** (*DocInput* **As DocInput**)

Parameters

DocInput

Object describing document input data for the current transfer.

Data Type: DocInput

Param: IN

Default Value: N/A

Comments

The DocInput event can be used in its basic form for notification of transfer progress, (for example, for updating a progress bar). The DocInput event is activated whenever the state of the DocInput object changes and whenever a block of data is ready to be transferred. The DocInput.BytesTotal, DocInput.BytesTransferred and DocInput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

You can also use the DocInput event to implement data streaming in pull mode. The PushStreamMode determines whether push mode or pull mode is used. By default, this property is False and pull mode is used. To use push mode, see the PushStreamMode property and the PushStream method.

To implement data streaming in pull mode, first call SendDoc. When calling SendDoc, the InputFile parameter should be omitted, and the DocInput.FileName and DocInput.DocLink properties should both be empty. This means that input data will be supplied by the user during DocInput event handling, as explained in the following steps.

After calling SendDoc, follow these steps during DocInput event handling. One step is performed each time an event is activated.

1. The first time the event is activated for a document transfer, the DocInput.State property will be set to icDocBegin (1). No action is necessary.
2. The second time the event is activated, the DocInput.State property will be set to icDocHeaders (2). The input document headers can be set at this time (if not specified previously) by modifying the DocInput.Headers collection.
3. For the next sequence of events, the DocInput.State will be set to icDocData (3). Call the DocInput.SetData method each time the event is activated as long as more data is available. This is how the next block of data is specified.
4. When no more data is available, simply do not call DocInput.SetData during event handling (when DocInput.State is set to icDocData (3)). This will signal the end of the transfer.
5. If an error occurs at any time, the DocInput.State property will be set to icDocError (4). The error information can be examined at this time by examining the Errors collection of the control. The control's standard Error event will be activated after the DocInput event.
6. The last time the event is activated (whether or not an error occurs), the DocInput.State property will be set to icDocEnd (5). No action is necessary.

Note: The first buffer of data may be supplied before calling SendDoc. This can be done by calling SetData before SendDoc or by passing the InputData argument to SendDoc. In this case, the DocInput event will not be activated to obtain the first block of data, and step 3 will occur when the second block of data is requested. If you do not respond to the DocInput icDocData event, only the initial block of data will be transferred. This transfers a single data buffer without using event handling.

Flow control for data streaming is handled via the Suspend property and Suspend method. The DocInput event will not be activated if the DocInput object is suspended.

DocOutput Object Overview

The DocOutput object describes output information for a document being transferred. It is the type of the DocOutput property, that is part of all controls with document output capabilities. In such controls, it is also an argument of the DocOutput event.

Properties, methods and events supported by the DocOutput Object are summarized alphabetically in the following table. For an explanation of the description categories, refer back to Chapter 1, section "Object Descriptions."

Property	Method	Event
AppendToFile	SetData	DocOutput
BytesTotal	GetData	
BytesTransferred	Suspend	
DocLink		
FileName		
Headers		
PushStreamMode		
State		
Suspended		

AppendToFile DocOutput Object Property

Description

Indicates whether the output file should be appended to or overwritten. If False, the existing output file will be overwritten.

Syntax

*object.***DocOutput.AppendToFile**

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

Boolean.

Default Value

True.

Range

True or False.

Comments

This property is only used when an output file is specified (e.g, when FileName is set or passed as a parameter in GetDoc).

BytesTotal DocOutput Object Property

Description

Total bytes to be transferred or zero, if not available.

Syntax

*object.***DocOutput.BytesTotal**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Long.

Default Value

Zero.

Range

>= zero

Comments

The BytesTotal property is available as soon as document transfer begins. The property value will not change until a new transfer is begun. This value may be zero if the size of the document is unknown. It may be set prior to the icDocEnd state once the final bytes transferred is known.

BytesTransferred DocOutput Object Property

Description

Number of bytes transferred so far.

Syntax

*object.***DocOutput.BytesTransferred**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Long.

Default Value

Zero.

Range

>= zero

Comments

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 DocOuput event is activated. 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).

DocLink DocOutput Object Property

Description

Used for data linking between two controls.

Syntax

*object.***DocOutput.DocLink**

Permission

R (Read Only).

Availability

R (Runtime).

Data Type

DocLink.

Default Value

Empty.

Range

Empty or a reference to another object’s DocInput.DocLink property

Comments

The DocLink property may be assigned before calling the GetDoc method. It should be set to the DocLink property of a DocInput object. This will cause data output from the DocOutput object to be used as the data input for the DocInput object.

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

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

FileName DocOutput Object Property

Description

Name of a local file containing the document to be retrieved.

Syntax

*object.***DocOutput.FileName** [**=** *String*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

Valid file name.

Comments

The FileName property may be set before calling the GetDoc method in a particular control or it may be passed as an argument to this method. If it is passed as an argument, the DocOutput.FileName property will be set to the argument value.

If the FileName property is not empty, data will be added to the file as it is received. If the FileName property is empty, no data will be written to an output file. However, data will be available via data linking and/or data streaming.

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

Headers DocOutput Object Property

Description

A collection of headers describing the current document being transferred.

Syntax

*object.***DocOutput.Headers**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

DocHeaders.

Default Value

N/A.

Range

N/A.

Comments

The contents of the Headers collection will be 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.

The Headers collection contains DocHeader items, each of which represents a MIME 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.

PushStreamMode DocOutput Object Property

Description

Indicates whether the stream is in push or pull mode. This property is set by the control.

Syntax

*object.***DocOutput.PushStreamMode**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Boolean.

Default Value

False.

Range

True or False.

Comments

The PushStreamMode of the DocOutput object is not normally needed by the user. However, it is provided for informational purposes. The DocOutput object does not have a PushStream method.

State DocOutput Object Property

Description

Indicates current state of the document retrieval.

Syntax

*object.***DocOutput.State**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

DocStateConstants

Default Value

icDocNone (0).

Range

The State property is maintained by the specific control. Each time it changes, the DocOutput 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)

Suspended DocOutput Object Property

Description

Indicates whether document transfer is currently suspended or not.

Syntax

*object.***DocOutput.Suspended**

Permission

R (Read Only).

Availability

R (Runtime).

Data Type

Boolean.

Default Value

False.

Range

True or False

Comments

The transfer is suspended if the Suspend method of the DocOutput object is called, or any DocInput object linked to it is suspended.

GetData DocOutput Object Method

Description

Retrieve the current block of data being received when the DocOutput event is activated.

Return Value

Void.

Syntax

*object.***DocOutput.GetData** *Data* [,*Type*]

Parameters

Data

Where retrieved data will be stored after the method returns successfully. If there is no data available for the requested type, *data* will be set to Empty.

Data Type: VARIANT

Param: OUT

Type

Optional. Type of data to be retrieved. The variant types that are supported are the same as listed in DocInput.SetData method.

Data Type: Variant.

Param: IN

Default Value: vbString

Comments

The GetData method should only be called during handling of the DocOutput event, when the State property is set to icDocData (3). In addition to using an output file (FileName property) and output link (OutputLink property), GetData may be called to process output data.

SetData DocOutput Object Method

Description

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

Return Value

Void.

Syntax

*object.***DocOutput.SetData** *Data*

Parameters

Data

Next block of data to be sent. For binary data, byte array should be used.

Data Type: VARIANT

Param: IN

Currently, the following variant types are supported.

Type	C++	VB 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

Comments

SetData may 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 will modify the data received by any DocInput objects that are 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.

Suspend DocOutput Object Method

Description

Suspends or resumes document transfer.

Return Value

Void.

Syntax

*object.***DocOutput.Suspend** *Suspend*

Parameters

Suspend

Indicates whether to suspend or resume transfer. If True, transfer is suspended. If False, transfer is resumed.

Data Type: Boolean

Param: IN

Comments

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.

DocOutput Object Events

The DocOutput object is not a control, therefore, it has no events. However, it is almost always associated with a control that has a DocOutput event. The DocOutput object is always a parameter of the DocOutput event, and the two may be used together to perform data streaming as well as to monitor progress of the document transfer.

DocOutput DocOutput Object Event

Description

Indicates that output data has been transferred or the DocOutput state has changed.

Syntax

*object*_**DocOutput** (*DocOutput* **As DocOutput**)

Parameters

DocOutput

Object describing document output data for the current transfer.

Data Type: DocOutput

Param: IN

Default Value: N/A

Comments

The DocOutput event can be used in its basic form for notification of transfer progress, (for example, for updating a progress bar). The DocOutput.BytesTotal, DocOutput.BytesTransferred and DocOutput.State properties can be examined to determine the current status of the transfer. This event can be ignored if no progress information is needed.

You can also use the DocOutput event for data streaming by examining the Headers collection and calling GetData when the DocOutput event is activated. The steps listed here describe the sequence of states in DocOutput event handling.

1. The first time the event is activated for a document transfer, the DocOutput.State property will be set to icDocBegin (1).
2. The second time the event is activated, the DocOutput.State property will be set to icDocHeaders (2). The input document headers can be examined at this time (or any later time) by examining the DocOutput.Headers collection.
3. For the next sequence of events, the DocOutput.State property will be set to icDocData (3). To process output data, call the DocOutput.GetData method each time the event is activated. An event is activated for each block of data transferred.
4. If an error occurs at any time, the DocOutput.State property will be set to icDocError (4). You can examine the error information at this time by examining the Errors collection of the control. The control's standard Error event will also be activated after the DocOutput event.
5. The last time the event is activated (whether or not an error occurs), the DocOutput.State property will be set to icDocEnd (5).

Flow control for data streaming is handled via the Suspend method and Suspended property. The DocOutput event will not be activated if the DocOutput object is suspended.

DocHeaders Collection Overview

The DocHeaders collection is used to access the MIME headers associated with a document. See DocHeader Item Overview for more information.

DocHeaders Collection supports the following properties and methods:

Property	Method
Count	Add
Text	Clear
	Item
	Remove

Count DocHeaders Collection Property

Description

The number of items in the collection.

Syntax

object.**Count**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

Zero.

Range

>=zero.

Text DocHeaders Collection Property

Description

Complete text of all headers in standard MIME header format.

Syntax

object.**Text** [= *String*]

Permission

W (Read/Write).

Availability

R (Runtime).

Data Type

String.

Default Value

None.

Range

N/A.

Comments

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.

Add DocHeaders Collection Method

Description

Adds a new item to the collection. The Name and Value parameters are converted to type String (BSTR) and become the Name and Value properties of the DocHeader item (described in later in this chapter).

Return Value

Void.

Syntax

object.**Add** *Name, Value*

Parameters

Name

Attribute name.

Data Type: VARIANT

Param: IN

Default Value: None

Value

Value for the specified attribute name.

Data Type: VARIANT

Param: IN

Default Value: None

Clear DocHeaders Collection Method

Description

Removes all items from the collection.

Return Value

Void.

Syntax

object.**Clear**

Parameters

None.

Item DocHeaders Collection Method

Description

Returns a Docheader item from the collection. The Item method is the default method for a collection. It is usually called implicitly when referencing the collection using an Index.

Return Value

Void.

Syntax

object.**Item** *Index*

Parameters

Index

Index may be either an integer or a string. Integer indices identify an item by its one-based index. String indices identify an item by its Name property. References by name are case-insensitive

Data Type: VARIANT

Param: IN

Default Value: None

Remove DocHeaders Collection Method

Description

Removes an item from the collection.

Return Value

Void.

Syntax

object.**Remove** *Index*

Parameters

Index

Number or name of the item to remove. Index may be either an integer or a string. Integer indices identify an item by its one-based index. String indices identify an item by its Name property. References by name are case-insensitive

Data Type: VARIANT

Param: IN

Default Value: None

DocHeader Item Overview

A DocHeader object is an item in a DocHeaders collection. DocHeader items represent individual name and value pairs in MIME headers.   The properties supported by the DocHeader item are:

Property

Name
Value

Name DocHeader Item Property

Description

The item name or MIME header label (not including the colon character). This property can be used as an identifier for items in the DocHeaders collection

Syntax

object.**Name** [= *string*]

Permission

W(Read/Write).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty string.

Range

N/A.

Value DocHeader Item Property

Description

The item value which in MIME headers is the text after the label, colon character, and any leading spaces.

Syntax

object.**Value** [= *string*]

Permission

W(Read/Write).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty string.

Range

N/A.

icErrors Collection Overview

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.

Protocol errors provide 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 will not 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.

The icErrors Collection supports the following properties and methods:

Property	Method
Count	Clear
Source	Item

Count icErrors Collection Property

Description

The number of items in the collection.

Syntax

object.**Count**

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

Zero.

Range

>=zero.

Source icErrors Collection Property

Description

The vbObject that the most recent error applies to, or vbEmpty. Implementation is control-dependent. Unless specified by the particular control documentation, the value for Source is vbEmpty.

Syntax

object.**Source**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

VARIANT.

Default Value

None.

Range

N/A.

Clear icErrors Collection Method

Description

Removes all items from the collection and resets the Source property to vbEmpty.

Syntax

object.**Clear**

Return Value

Void.

Parameters

None.

Item icErrors Collection Method

Description

Returns an item from the collection.

Syntax

object.**Item** *Index*

Return Value

icError

Parameters

Index

Number or name of item to be returned. Index may be either an integer or a string. Integer indices identify an item by its one-based index. String indices identify an item by its Name property. References by Name are case-insensitive

Data Type: VARIANT

Param: IN

Default Value: None

Comments

The Item method is the default method for a collection.

icError Item Overview

An icError object is an item in an icErrors collection containing error messages. The following is a list of the properties supported by the icError item.

Property

Code
Description
Type

Code icError Item Property

Description

Integer error code for the given error type.

Syntax

object.**Code**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Long.

Default Value

0.

Range

0-32767.

Description icError Item Property

Description

Text description of the error.

Syntax

object.**Description**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty string.

Range

N/A.

Type icError Item Property

Description

String label for the type of error, with standard (predefined) labels such as “protocol” and “transport”. Individual controls may have additional Type labels.

Syntax

object.**Type**

Permission

R (Read only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty string.

Range

N/A.

Glossary

A
access
account
ANSI
API
application
ARP
ASCII
asynchronous

B
buffer

C
CGI
client

D
daemon
DLL
default
DNS
DocStream
domain
driver

E
Ethernet
event message
export

F
FDDI
file access
file server
Finger
FTP

G
group ID

H
host
HTML
ICMP

I, J, K
Internet
Intranet
IP

L
log in

log out

M
mail server
MAC
MIB
MIME

N
NDIS
network address
network installer
NEWT
NEWT Intranet ActiveX
NIC
NIS
NMMODEM
NNTP

O
OLE

P, Q
POP
prompt
protocol

R
remote
remote host
reply code
result code
RFC
RPC

S
script
server
socket
SMTP
SNMP
subnet

T
TCP/IP
Telnet
terminal emulator
TFTP
token ring
TSR

U, V
UDP
user ID
user name

access
Entry to or communication with a particular object, such as an operating system, specific files or accounts.

account
An entity that is established as an authorized user of the system.

ANSI

The American National Standards Institute sets standards for the U.S. computer industry. ANSI participates in defining network protocol standards.

API

Application Program Interface. A standard interface that allows upper layer applications to work with different communication protocol stacks.

application

An application is a computer program that performs a certain task. FTP, Telnet, and TN3270 are some of the applications provided by NetManage.

ARP

Address Resolution Protocol. The TCP/IP protocol used to dynamically bind a high level IP address to a low-level physical hardware address. ARP is only across a single physical network and is limited to networks that support hardware broadcast.

ASCII

The American Standard Code for Information Interchange, which is widely accepted code for representing alphanumeric information.

asynchronous
A non-blocking method of data transfer where transferred data does not arrive immediately.

buffer

A temporary storage area for data during the transfer of that data between the computer and a peripheral or between parts of a computer to prevent loss of information.

CGI

Common Gateway Interface. A standard used for client applications to access information servers (such as HTTP or Web servers). CGI programs are executed in real-time to output dynamic information.

client

A client is a computer application that uses resources provided by another machine on the network. Most of NetManage's applications can run as both client and server. It is the active partner in a client/server relationship.

daemon
An application, running on a system, that provides network resources to client applications, either local or remote.

DLL
Dynamic Link Library.   A method of sharing code between applications.

default
A value supplied by the system when a user does not specify a required command, parameter or qualifier.

DocStream

An architecture for controlling data transfer.   Every control based on DocStream architecture is either data target or data source.

domain

A named group of machines on the network. A domain name consists of a sequence of names (labels) separated by periods (dots).

DNS
Domain Name Server. An on-line distributed database that maps machine names into IP addresses or vice-versa.

driver

A software module that controls an input/output port or external device such as a keyboard or a monitor. TCP/IP uses a driver to control the network interface cards.

Ethernet
Ethernet is a type of network that supports high-speed communication among systems.

event message
An event message is a response to a function call or a command request from a client.

export
Export is the process that makes a function available so that other applications can access it.

FDDI
Fiber Distributed Data Interface. A network standard used with fiber-optic backbones.

file access
File access allows users to work with a remote file as if the file is local.

FTP

File Transfer Protocol.   The FTP application is used to provide file transfer services across a wide variety of systems through the use of the File Transfer Protocol (FTP). Usually implemented as application level programs, FTP uses the Telnet and TCP protocols. The server side may require a client to supply a login identifier and password before it honors requests.

file server
A process running on a computer that provides access to files on that computer to programs running on remote machines.

Finger
A standard protocol that lists information about users on another host.

group ID
A unique number associated with each group name on the server.

host
Any end user computer system that connects to a network. Hosts range in size from personal computers to supercomputers.

HTML

Hypertext Markup Language.   A standard format used to place hypertext documents on line for access from the WebSurfer.

ICMP

Internet Control Message Protocol. The ICMP delivers error and control messages from hosts to the requesters. An ICMP test can determine whether a destination is reachable and responding.

Internet

When capitalized, the world-wide network of networks connected to each other using the IP and other similar protocols. The Internet provides file transfer, remote login, electronic mail and other services. When not capitalized, any collection of distinct networks working together as one.

Intranet

A private enterprise network that uses TCP/IP standards-based networking technologies for host access, workgroup collaboration, desktop and network resources management, and developer tools for custom applications to maximize the enterprise's productivity. For example, Web began as an Internet application and has now been incorporated into internal company applications.

IP

The TCP/IP standard protocol that defines the IP as a unit of information passed across an Internet and provides the basis for packet delivery service.   IP includes the ICMP control and error message protocol as an integral part. The entire protocol suite is often referred to as TCP/IP because TCP and IP are the two most fundamental protocols.

log in

To perform a sequence of actions at terminal that establishes a user's communication with the operating system and sets up default characteristics for the user's terminal session.

log out
To terminate interactive communication with the operating system, and end a terminal session.

mail server

A host and its associated software that offer electronic mail reception and forwarding service. Users may send messages to, and receive messages from any other use in the system.

MAC

A MAC layer transmission mechanism interface is used for adding custom protocols to the communication stack. Any incoming packets utilizing protocols currently not supported will be routed to this socket, SOCK_MAC.

MIB

Management Information Base. The set of variables that a gateway running SNMP maintains. MIB-II refers to an extended management database that contains variables not shared by SNMP.

MIME

Multipurpose Internet Mail Extension. MIME defines the format of message bodies to allow multi-part textual and non-textual message bodies to be represented and exchanged without loss of information.

NDIS

Network Device Interface Specification. The NDIS specification is used for all communication with network adapters. The specification was developed by Microsoft and 3COM to provide a common programming interface for MAC drivers and transport drivers. NDIS works primarily with LAN manager and allows multiple protocol stacks to share a single network interface card.

network address
A unique number associated with a host that identifies it to other hosts during network transactions.

network installer

Network Chameleon is a complete set of TCP/IP utilities that can be installed and run from a central network file server.

NEWT
NetManage Enhanced Windows TCP/IP.   NEWT is a TCP/IP communication stack for Microsoft Windows.

NEWT Intranet ActiveX

A technology that lets an expert Internet programmer develop complete applications for the Internet. Reusable component of Microsoft architecture now known as ActiveX.

NIC

Network Information Center. The NIC at SRI International in Menlo Park, California, assigns IP addresses and network numbers per request submitted by an organization. The number assigned is appropriate to the number of host devices on the network.

NIS

Network Information Service.   The Network Information Service (NIS) was formerly known as the Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has been changed.

NMMODEM

NMMODEM maintains the NetManage Modems Library. The library includes the default settings for more than 250 of the most common models of modems. This API provides functions for selecting, getting, and changing modem commands and parameters.   It also allows new custom defined modems to be added to the library.

NNTP
Network News Transfer Protocol. Used to access NNTP news servers.

OLE

Object Linking and Embedding. This technology defines a set of standard interfaces that allow functions from one application to access the services of another application.

POP

Word or words used by the system to assist a user's response. Such messages generally ask users to respond by typing some information in the following field.

prompt

Word or words used by the system to assist a user's response. Such messages generally ask users to respond by typing some information in the following field. Post Office Protocol. This protocol is used by the Mail application to provide electronic mail services.

protocol
Rules defining how two entities, such as client and server, should communicate with each other.

RFC

Request for Comment. The RFC documents describe all aspects and issues associated with the Internet protocols.

RPC

Remote Procedure Call. A mechanism defined by SUN Microsystems that provides a standard for initiating and controlling processes on remote or distributed computer systems.

remote
Files, devices and users not attached to your local machine.

remote host
The computer receiving the network command.

reply code
A reply code is a response code sent from the host.

result code
A result code indicates the status of function call unless otherwise specified or the status of an event.

script
A sequence of ASCII text lines that are stored in a file.

server
A computer that provides services to a network.

socket
Pairing of an IP address and a port number.

SMTP
Simple Mail Transfer Protocol.   A protocol used by the Mail application to provide electronic mail services.

SNMP

Simple Network Monitoring Protocol. A standard protocol used to monitor network activity on agent nodes from management stations.

subnet
Subnet is a field used by routers and hosts for routing packages on the network.

TCP/IP

Transmission Control Protocol/Internet Protocol (TCP/IP). TCP allows a process on one machine to send data to a process on another machine using the IP protocol. TCP is a connection-oriented protocol and can be used as a full duplex or one-way simplex connection.

Telnet

The Telnet application provides virtual terminal services for a wide variety of remote system using the Telnet protocol. The application allows a user at one site to interact with a remote system at another sites as if the user's terminal is connected to the remote machine.

TFTP

Trivial File Transfer Protocol. TFTP is a standard TCP/IP protocol that allows simple file transfer to and from a remote system without directory or file listing. TFTP is used where FTP is not available.

terminal emulator

A program that makes a PC screen and keyboard act like a video display terminal of another computer. Current emulations supported by NetManage are ANSI, VT52, VT100, VT220, TN3270, and TN5250.

TSR

Terminate-and Stay-Resident. A DOS program that is loaded into memory before Windows and stays in memory until the machine is rebooted.

token ring

Token ring is a type of ring-shaped network that supports high-speed communications between computers. A distinguishing packet, called a token, is transferred from machine to machine. Only the machine that holds the token can transmit the packet.

UDP
User Datagram Protocol.   A transport protocol that offers a connectionless-mode transport service in the Internet suite of protocols.

user ID
A unique number, created by your system, that is associated with each user name on a server system.

user name
A character string, usually assigned by the system administrator that identifies a user on the system.

utility
A command or operation that works at the level of the operating system.

WinHLLAPI

WinHLLAPI is a high level language application program interface that you can use to program in high level languages such as COBOL, Pascal, BASIC, or C that is compatible with Windows.

WinSock
Standard Communication Protocol for Internet Networking.

Error Codes and Messages

This Help file lists the error codes and messages for all ActiveX controls (formerly OLE Controls). Error messages which may apply to more than one ActiveX control are referred to as Common Errors and are in the range of 1000-2000.

To help you locate the error code quickly, the following ranges have been assigned for control specific errors.

Range	Control
2100-2199	FTP
2200-2299	NNTP
2300-2399	SMTP
2400-2499	POP
2500-2599	WCGI
2600-2699	HTML
2700-2799	HTTP
3101-3105	MIME

Common Errors

The error codes and messages in this category relate to more than one ActiveX control.

Error Code	Error Message
1001	Error sending
1002	Error receiving
1003	Error connecting
1004	Error disconnecting
1005	Wrong protocol or connection state for the requested transaction or request
1006	Error when parsing; data supplied by the server is of unexpected format
1007	Early close issued which was not expected
1008	Busy; an action was requested that cannot be completed because protocol instance is busy waiting for a response
1009	Unknown error
1010	Internal error
1011	Timed-out; response to a request or notification about an event was not received in the expected time span
1012	Out of Memory
1013	The argument passed to a function was not in the correct format or in the specified range
1014	The protocol reply to a request indicates that there was an error in the reply
1015	Authentication failed
1016	The connection is busy waiting for a reply from the server

1100	Successful
1101	Unsupported variant types
1102	Invalid URL: URL not recognized
1103	Invalid operation at current state
1104	Argument is out of range
1105	Property cannot be set in the current protocol or connection state
1902	File or directory not found
1913	Permission to access this file denied
1917	Cannot create file that already exists
1924	Too many files already open
1928	Not enough space on disk (disk full)
1922	Invalid path name or other argument

FTP Error Codes

The following error codes apply only to the FTP ActiveX Control.

Error Code	Error Message
2104	Port Command Failed. Unable to open Port.
2105	Abort Command Failed. Unable to Abort last command.
2106	Account Command Failed. Unable to complete.
2107	Change Directory Command Failed. Unable to change to specified directory.
2108	Connect Command Failed. Unable to connect to remote host.
2109	Create Directory Command Failed. Unable to create specified directory.
2110	Delete Directory Command Failed. Unable to delete specified directory.
2111	Delete File Command Failed. Unable to delete specified file.
2112	Disconnect Command Failed. Unable to disconnect from remote host.
2113	Get File Command Failed. Unable to retrieve specified file.
2114	Help Command Failed. Unable to retrieve help from remote host.
2115	NOOP Command Failed. Control connection error.
2116	Name List Command Failed. Unable to retrieve Named list; possible data connection error.
2117	List Command Failed. Unable to retrieve detailed list; possible data connection error.
2118	Parent directory Command Failed. Unable to change directory up.
2119	Print Directory Command Failed. Unable to print current directory of remote host.
2120	Put File Command Failed. Unable to put file on remote host.
2121	Put Unique File Command Failed. Unable to put unique file on remote host.

2122	Reinitialize Command Failed. Unable to reinitialize login on remote host.
2123	Rename File Command Failed. Unable to rename specified file on remote host.
2124	Retrieve File Command Failed. Unable to retrieve specified file from remote host.
2125	Status Command Failed. Unable to retrieve status from remote host.
2126	System Command Failed. Unable to issue SYST command to remote host.
2127	Type Command Failed. Unable to set transfer type on remote host.
2128	Error setting OutputDocStream property.
2129	Error setting InputDocStream property.
2130	No error detected.
2131	Firewall host could not complete the OPEN command.
2132	Firewall host rejected the username.
2133	Authentication with the firewall host failed.
2134	Firewall host could not complete the SITE command.
2135	Firewall host could not complete the USER user@host command.
2157	Command not implemented.
2171	Maximum connection(<number>) reached.

NNTP Error Codes

The following error codes apply only to the NNTP ActiveX Control.

Error Code	**Error Message**
2203	NNTP server does not allow posting.

SMTP Error Codes

The following error codes apply only to the SMTP ActiveX Control.

Error Code	**Error Message**
2302	Can't create temporary mail file
2303	Unable to send mail

POP Error Codes

The following error codes apply only to the POP ActiveX Control.

Error Code	**Error Message**
2450	RetrieveMessage Command Failed. Unable to retrieve message.
2451	Delete Command Failed. Unable to delete message.
2452	Reset Command Failed. Unable to unmark deleted message(s).
2453	Last Command Failed. Unable to find the highest message number accessed by client.
2454	RefreshMessageCount Command Failed. Unable to ascertain the number of messages marked as deleted.
2455	Noop Command Failed. Unable to test the connection.
2456	Quit Command Failed. Error while quitting.

2457	TopMessage Command Failed. Unable to retrieve the TopLines of the message.

CGI Error Codes

For this release there are no HTTP error codes.

MIME Error Codes

The following error coded apply only to the MIME ActiveX Control.

Error Code	**Error Message**
3101	Encoding format is required for icOtherMIMEType. Operation cancelled.
3102	Unrecognized Encoding Format. Operation cancelled.
3103	Unrecognized MIME type. Operation cancelled.
3104	Missing or invalid source file. Operation cancelled.
3105	Unable to create output file. Operation cancelled.

HTTP Error Codes

For this release there are no HTTP error codes.

HTML Error Codes

For this release there are no HTML error codes.

WinSock Error Codes

The following error codes apply to the WinSock ActiveX Controls.

Error Code	Error Message
10004	The operation is canceled.
10013	The requested address is a broadcast address, but flag is not set.
10014	Invalid argument.
10022	Socket not bound, invalid address or listen is not invoked prior to accept.
10024	No more file descriptors are available, accept queue is empty.
10035	Socket is non-blocking and the specified operation will block.
10036	A blocking Winsock operation is in progress.
10037	The operation is completed. No blocking operation is in progress.
10038	The descriptor is not a socket.
10039	Destination address is required.
10040	The datagram is too large to fit into the buffer and is truncated.
10041	The specified port is the wrong type for this socket.
10042	Option unknown, or unsupported.
10043	The specified port is not supported.
10044	Socket type not supported in this address family.
10045	Socket is not a type that supports connection oriented service.
10047	Address Family is not supported.
10048	Address in use.
10049	Address is not available from the local machine.
10050	Network subsystem failed.
10051	The network cannot be reached from this host at this time.
10052	Connection has timed out when SO_KEEPALIVE is set.
10053	Connection is aborted due to timeout or other failure.
10054	The connection is reset by remote side.
10055	No buffer space is available.
10056	Socket is already connected.
10057	Socket is not connected.
10058	Socket has been shut down.
10060	The attempt to connect timed out.
10061	Connection is forcefully rejected.
10201	Socket already created for this object.
10202	Socket has not been created for this object.
11001	Authoritative answer: Host not found.
11002	Non-Authoritative answer: Host not found.
11003	Non-recoverable errors.
11004	Valid name, no data record of requested type.

Redistribution of ActiveX Controls

After developing an application(s), you may want to distribute it to others. In order for the application to work on other machines you will have to include some of the SDK within your installation.

There are two parts to the redistribution files. Some of the files that must be redistributed are redistributable Microsoft files, and some of the files are NetManage files. Click on the button below to see a table showing the dependencies.

{button Distribution Files,JI(`NIA.HLP',`IDH_Distributable_Files')}

Distributable Files

ActiveX Control	NetManage Required DLLs	Microsoft Redistributable DLLs
FTPCT.OCX	NMOCOD.DLL, NMORENU.DLL, NMSCKN.DLL, NMFTPSN.DLL	OLEPRO32.DLL
HTML.OCX	NMOCOD.DLL, NMORENU.DLL, NMSCKN.DLL, NMW3VWN.DLL	OLEPRO32.DLL, CTL3D32.DLL, MSVCRT40.DLL, VB40032.DLL
HTTPCT.OCX	NMOCOD.DLL, NMORENU.DLL, NMSCKN.DLL	OLEPRO32.DLL
HTTPSR.OCX	NMOCOD.DLL, NMORENU.DLL, NMSCKN.DLL	OLEPRO32.DLL
INETCT.OCX	NMOCOD.DLL, NMORENU.DLL, NMSCKN.DLL	OLEPRO32.DLL
INETSR.OCX	NMOCOD.DLL, NMORENU.DLL, NMSCKN.DLL	OLEPRO32.DLL
MIME.OCX	NMOCOD.DLL, NMORENU.DLL, NMSCKN.DLL	OLEPRO32.DLL
NNTPCT.OCX	NMOCOD.DLL, NMORENU.DLL, NMSCKN.DLL	OLEPRO32.DLL
POPCT.OCX	NMOCOD.DLL, NMORENU.DLL, NMSCKN.DLL	OLEPRO32.DLL
SMTPCT.OCX	NMOCOD.DLL, NMORENU.DLL, NMSCKN.DLL	OLEPRO32.DLL
WCGI.OCX	NMOCOD.DLL, NMORENU.DLL, NMSCKN.DLL	OLEPRO32.DLL
WINSCK.OCX	NMOCOD.DLL, NMORENU.DLL, NMSCKN.DLL	OLEPRO32.DLL

Installation Process

To install your package with support for NetManage ActiveX controls on a target system, follow the procedures below.

Step1. Search for previously installed versions.

Your installation process should search for previous versions of the controls on the target machine. The purpose of this is to verify that the software you are installing is the latest.

The NetManage and Microsoft files should be found in the Windows system directory of the target machine. For Windows 95, this directory would be %windir%\system. For Windows NT, this directory would be %windir%\system32.

You can use Microsoft's **VerFindFile()** function to determine if the ActiveX files exist on the target machine.

If files are found, go on to Step 2; otherwise, go to Step 3.

Step 2. Compare file versions

If step 1 found some of the NetManage files to installed already on the target machine, you will need to check the version of both the files you are installing and the files already on the machine. We suggest you replace the files on the target machine if the files you are installing are newer.

Microsoft's **VerInstallFile()** function can be used to install files based on the version information. This function checks the file version and can also install the file.

If the files on the target machine are older, go to Step 3; otherwise, go to Step 4.

Step 3. Install files

As described in Step 1, the NetManage redistributable files should be installed in the Windows system directory.

Step 4. Registration

You must register the common DLL for the ActiveX controls. After this file is registered, you may register some or all of the controls, depending on your applications requirements.

To register these controls, run the following commands from your installer.

- REGSVR32 /s NMOCOD.DLL
- REGSVR32 /s FTPCT.OCX
- REGSVR32 /s HTML.OCX
- REGSVR32 /s HTTPCT.OCX
- REGSVR32 /s HTTPSR.OCX
- REGSVR32 /s INETCT.OCX
- REGSVR32 /s INETSR.OCX
- REGSVR32 /s MIME.OCX
- REGSVR32 /s NNTPCT.OCX
- REGSVR32 /s POPCT.OCX
- REGSVR32 /s SMTPCT.OCX
- REGSVR32 /s WCGI.OCX
- REGSVR32 /s WINSCK.OCX

You must run REGSVR32 on NMOCOD.DLL before you register any other OCX. You need not register more OCXs than those used in your control. For example, if you only used FTPCT.OCX, you only need to register NMOCOD.DLL and FTPCT.OCX.

Compatibility Issues

Please be aware that Chameleon applications utilize NetManage ActiveX OCXs and DLLs, and that the Chameleon installer will update these OCXs and DLLs in the system directory. It is also possible that other vendors who utilize NetManage ActiveX controls will have already installed an older version of the controls on the target system.

While every effort will be made to ensure compatibility between releases of Chameleon and releases of NetManage ActiveX, it is possible that newer controls will disable or change execution characteristics of your ActiveX application. For example, this may cause you to break someone else's application by installing newer controls or vice versa).

You should be aware that there may be incompatibilities between different NetManage Chameleon 6.0+ releases and the NMW3VWN.DLL and NMOCOD.DLL shipped with other products (such as Delphi, Internet Control Pack Beta, and the shipping version of the NetManage NEWT Intranet ActiveX Controls (6.02))

ActiveX Removal Instructions

If you provide your customer with an uninstall program, that program must interrogate the following files for usage counts prior to removing the files from the user's system. If the usage count indicates that the file still has users, do not remove it. It's probably a good idea to maintain usage counts for any files you install for your application.

ActiveX Files

- MSVCRT40.DLL
- OLEPRO32.DLL
- REGSVR32.EXE
- CTL3D32.DLL
- VB40032.DLL

- NMFTPSN.DLL
- NMSCKN.DLL
- NMORENU.DLL
- NMW3VWN.DLL
- NMOCOD.DLL

- FTPCT.OCX
- HTML.OCX
- HTTPCT.OCX
- HTTPSR.OCX
- INETCT.OCX
- INETSR.OCX
- MIME.OCX
- NNTPCT.OCX
- POPCT.OCX
- SMTPCT.OCX
- WCGI.OCX
- WINSCK.OCX

File Descriptions

NetManage ActiveX DLLs

NMSCKN.DLL	This DLL provides a TCP/IP WinSock layer to the ActiveX controls.
NMOCOD.DLL	This DLL provides docstreaming and other various shared code.
NMORENU.DLL	This DLL provides the language resources (for English).
NMFTPSN.DLL	This DLL provides FTP support for the FTP ActiveX control.
NMW3VWN.DLL	This DLL provides HTML viewing support for the HTML ActiveX control.

NetManage ActiveX Controls

FTPCT.OCX	This control provides FTP client support.
HTML.OCX	This control provides HTML viewer support.
HTTPCT.OCX	This control provides HTTP client support.
HTTPSR.OCX	This control provides HTTP server support.
INETCT.OCX	This control provides generic Internet client support.
INETSR.OCX	This control provides generic Internet server support.
MIME.OCX	This control provides MIME encode/decode support.
NNTPCT.OCX	This control provides NEWS/NNTP client support.
POPCT.OCX	This control provides POP3 client support.
SMTPCT.OCX	This control provides SMTP client support.
WCGI.OCX	This control provides WCGI support.
WINSCK.OCX	This control provides TCP and UDP support.

Microsoft Files

CTL3D32.DLL	This DLL provides MFC 3D control support.
MSVCRT40.DLL	This DLL provides C run-time library support.
OLEPRO32.DLL	This DLL provides OLE property frame and standard types support.
REGSVR32.EXE	This application provides an interface to register controls.
VB40032.DLL	This DLL is the Visual Basic runtime DLL.

Important Licensing Information

When you purchase [license?] the ActiveX toolkit from NetManage, the licensing agreement restricts you from distributing these tools to your end users. Therefore, you must not ship the file LICENSES.REG with the application. This will enable your end users to develop applications using the ActiveX controls, thereby breaking your licensing agreement with NetManage. This can have legal consequences.

If you wish to distribute your application with the ActiveX controls and the LICENSES.REG file, please contact the Legal Department at NetManage.

Troubleshooting

Missing DLLs

If your application fails to run on the target machine, you may want to check the success of the registration of the ActiveX controls. REGSVR32 will attempt to tell you what DLLs are missing if any.

In a DOS window, type:

REGSVR32 <ActiveX Control Name>.OCX

This program will attempt to determine what is wrong with the loading of the ActiveX control and a Message Box will be display describing the reason it was unable to load the control. It may be a missing DLL error or some other fault.

If the ActiveX control loads correctly, the problem may lay within your application.

Problems due to Beta versions:

If your installation program detects an old version of the Microsoft Internet Control Pack(ICP), a BETA version of the Netmanage ActiveX controls, or NetManage ActiveX control set installed from another vendor, such as Borland's Delphi your installation should to clean up the registry using the NMCLN.EXE application. Beta versions used different GUIDs in the Windows Registry for the ActiveX controls.

All of the controls released via these channels are stamped with a version key in the resource that is consistent over all the controls and their support DLL's. However, early releases of ActiveX utilized different GUIDS in the system registry. If these GUIDS remain on the system it may affect the installation of newer versions of ActiveX.

To correct this situation, you should run the utility clean program, NMCLN.EXE , prior to control registration on the target system. The syntax you must use is 'NMCLN GUIDS.TXT'. This utility returns no information and will only remove the old style NetManage ActiveX GUIDS from the system registry if they are found.

Your installer must run this utility prior to registering the ActiveX controls. Once this utility has been run, you may remove it from the target system. This utility runs without a visible interface, so you may run it silently from your installer.

Example:

```
NMCLN GUIDS.TXT
REGSVR32 /S NMOCOD.DLL
REGSVR32 /S FTPCT.OCX
DELETE NMCLN.EXE
DELETE GUIDS.TXT
```

NEWT Intranet ActiveX Online Help

This is the main document for the NEWT Intranet ActiveX online Help system. This is the primary Help document, which contains the template ROBOHELP.DOT. This document is not displayed in the Help window, but simply stores project information.

All other files in this project use the template ROBORTF.DOT. These files are:

- AX1_Intro.doc
- AX2_Comn.doc (Common Control Objects)
- AX3_CGI.doc
- AX4_FTP.doc
- AX5_HTML.doc
- AX6_HTTP.doc
- AX7_INET.doc
- AX8_MIME.doc
- AX9_NNTP.doc
- AX10_POP.doc
- AX11_SMTP.doc
- AX12_TCP.doc
- AX13_UDP.doc
- AXR_DIST.doc
- Glossary.doc
- ErrorCode.doc

The Help project file is NIA.HPJ.
The Help directory is NIA.

The graphics directory is c:\NIA\graphics.

Context-Sensitive Help IDs

I have created an alias file that links each control to the appropriate Overview Help topic for each control.

Control Name	Program-ming ID	Help Topic ID
CGI	3001000	CGI ActiveX Control Overview
FTP Client	2905052	FTP Client ActiveX Control Overview
HTML	2902062	HTML ActiveX Control Overview
HTTP Client	2908010	HTTP Client ActiveX Control Overview
HTTP Server	3002000	HTTP Server ActiveX Control Overview
Internet Client	3003008	Internet Client ActiveX Control Overview
Internet Server	3004000	Internet Server ActiveX Control Overview
MIME	3001100	MIME (with UUEncode) ActiveX Control Overview
NNTP Client	2909028	NNTP Client ActiveX Control Overview
POP Client	2906019	POP Client ActiveX Control Overview
SMTP Client	2907004	SMTP Client ActiveX Control Overview
Winsock Controls	2901033	WinSock ActiveX Controls (Created topic in TCP that points to both)
Winsock TCP	2901034	WinSock TCP ActiveX Control

		Overview
Winsock UDP	2901037	WinSock UDP ActiveX Control Overview