NETMANAGE

Common Control Objects

The following Help Topics are available:

For Help on Help, Press F1

Net*Manage*

NetManage develops, markets and supports an integrated set of TCP/IP inter-networking applications and development tools for Microsoft Windows. NetManage software facilitates communication, productivity and the administration of personal computers across dissimilar networking environments. The Company's award-winning product families include Chameleon and ECCO.

The company is located at 10725 North De Anza Blvd. Cupertino, CA 95014, USA

Phone: 408-973-7171 Fax: 408-973-8272.

International phone: +972-4-8550234 Fax +972-4-8550122

General

A number of objects are common to all ActiveX Controls. These objects form a basis for the design of the client protocols. The Internet Solutions Pack is designed as a family of eight controls with a few common interfaces that allow you to develop applications quickly. Famliarizing yourself with these common interfaces can save you time later.

The common objects are:

- DocInput
- DocOutput
- DocHeaders Collection
- icErrors Collection

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 is a collection with its own set of properties and methods.

Naming Conventions

Objects described in the Properties, Methods and Events section are preceded by the required parameter: object. During execution object translates to the name of the relative control or collection name.

When using a collection, object becomes the collection name. For example, icErrors collection objects are preceded by icErrors.

DocInput Object

The DocInput object describes input information for a document being transferred. It is the type of the DocInput property that is part of all controls with document input capabilities. In such controls, it is also an argument of the DocInput event.

The DocInput object has its own set of properties and methods The DocInput event is documented here even though it is an event of the particular control, not of the DocInput object itself.

Properties

Properties related to the DocInput object are.

BytesTotal

BytesTransferred

DocLink

FileName

Headers

PushStreamMode

State

Suspended

BytesTotal

Description

Total bytes to be transferred or zero, if not available.

Syntax

*object.***DocInput.BytesTotal**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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.

BytesTransferred

Description

Number of bytes transferred so far.

Syntax

*object.***DocInput.BytesTransferred**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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 or 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

Description

Copy of a DocOutput.DocLink property when data linking is used or empty if data linking is not used.

Syntax

*object.***DocInput.DocLink** [= *String*]

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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 used as the data input for the DocInput object.

Input data 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

Description

Name of a local file containing the document to be transferred.

Syntax

*object.***DocInput.FileName** [**=** *String*]

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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 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

Description

A collection of headers describing the current document being transferred.

Syntax

*object.***DocInput.Headers**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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 GetDoc or SendDoc method of the specific control or it may be passed as an argument to these methods. If it is passed as an argument, the items in the DocInput.Headers collection will be replaced with those in the collection specified in the argument.

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

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*]

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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) and data is specified when the DocInput event is activated.
- Set the PushStreamMode property to True before initiating the document transfer. See the PushStream method for more information on this technique.

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

Description

Indicates current state of the document transfer.

Syntax

*object.***DocInput.State**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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

Description

Indicates whether document transfer is currently suspended or not.

Syntax

*object.***DocInput.Suspended**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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.

Methods

Methods are called to perform a particular operation The methods performed by the DocInput Object are.

GetData

PushStream

SetData

Suspend

GetData

Description

Retrieve the current block of data to be transferred when the DocOutput event is activated.

Return Value

Void.

Syntax

*object.***DocInput.GetData** *Data* [,*Type*]

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

Parameters

Data

Stores retrieved data after the method returns successfully. If there is not enough data available for 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: Long
Param: IN
Default Value: vStringing

Comments

The GetData method can only 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

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**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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 specified when the DocInput event is activated. You should not call PushStream.
- PushStreamMode is set to True, and when data is available you call the PushStream method. PushStream is called to perform the next step of the document transfer. PushStream changes the State property based on the next step of the transfer, activates the DocInput event as needed, and returns to wait for the next call to PushStream.

  When using this technique, instead of setting document information in the DocInput event handler, you can set document information before calling PushStream. For example, when transferring data, the SetData method may be called before calling PushStream.

  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

Description

Specify the next data buffer to be transferred when the DocInput event is activated.

Return Value

Void.

Syntax

*object.***DocInput.SetData** *Data*

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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++
Byte	VT_UI1
Integer	VT_I2
Long	VT_I4
Single	VT_R4
Double	VT_R8
Currency	VT_CY
Date	VT_DATE
Boolean	VT_BOOL
SCODE	VT_ERROR
String	VT_STRING
Byte Array	VT_ARRAY | VT_UI1

Comments

SetData is normally 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. Calling SetData in these cases will modify the data transferred to the target document.

Suspend

Description

Suspends or resumes document transfer.

Return Value

Void.

Syntax

*object.***DocInput.Suspend** *Suspend*

The object placeholder is required and evaluates to the name of the relevant object or collection during object during execution.

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.

Events

The DocInput object is not a ActiveX 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

Description

Indicates that input data has been transferred.

Syntax

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

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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 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.
1. 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.
1. 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.
1. 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.
1. 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.
1. 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

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.

This object has its own set of properties, methods and events.

Properties

Properties supported by the DocOutput object are.

BytesTotal

BytesTransferred

DocLink

FileName

Headers

PushStreamMode

State

Suspended

BytesTotal

Description

Total bytes to be transferred or zero, if not available.

Syntax

*object.***DocOutput.BytesTotal**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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.

BytesTransferred

Description

Number of bytes transferred so far.

Syntax

*object.***DocOutput.BytesTransferred**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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 or 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

Description

Assigns the DocInput.DocLink property when data linking is used.

Syntax

*object.***DocOutput.DocLink**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

Permission

R (Read Only).

Availability

R (Runtime).

Data Type

DocLink.

Default Value

Empty.

Range

Object reference

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

Description

Name of a local file containing the document to be transferred.

Syntax

*object.***DocOutput.FileName** [**=** *String*]

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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 appended to the file as it is transferred. 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

Description

A collection of headers describing the current document being transferred.

Syntax

*object.***DocOutput.Headers**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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

Description

Indicates whether the stream is in push or pull mode. This property is set by the control.

Syntax

*object.***DocOutput.PushStreamMode**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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.

For more information, see the PushStreamMode property of the DocInput object for more information.

State

Description

Indicates current state of the document transfer.

Syntax

*object.***DocOutput.State**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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

Description

Indicates whether document transfer is currently suspended or not.

Syntax

*object.***DocOutput.Suspended**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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.

Methods

Methods are called to perform a particular operation performed by the DocOutput Object. The following section describes the methods

GetData

SetData

Suspend

GetData

Description

Retrieve the current block of data being transferred when the DocOutput event is activated.

Return Value

Void.

Syntax

*object.***DocOutput.GetData** *Data* [,*Type*]

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

Parameters

Data

Where retrieved data will be stored after the method returns successfully. If there is not enough data available for requested type, *data* will be set to Empty.
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: Long.
Param: IN
Default Value: vStringing

Comments

The GetData method may only be called during handling of the DocOutput event, when the State property is set to DATA (3). In addition to using an output file (FileName property) and output link (OutputLink property), GetData may be called to process output data.

SetData

Description

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

Return Value

Void.

Syntax

*object.***DocOutput.SetData** *Data*

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

Parameters

Data

Next block of data to be sent. For binary data, byte array should be used.
Data Type: VARIANT
Param: IN

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

Description

Suspends or resumes document transfer.

Return Value

Void.

Syntax

*object.***DocOutput.Suspend** *Suspend*

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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.

Events

The DocOutput object is not an OCX, 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

Description

Indicates that output data has been transferred.

Syntax

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

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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. One step occurs each time the event if activated.

1. The first time the event is activated for a document transfer, the DocOutput.State property will be set to icDocBegin (1).
1. 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.
1. 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.
1. 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.
1. 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

The DocHeaders collection is used to access the MIME headers associated with a document. See the Headers property of the DocInput and DocOutput object for more information.

Properties

Methods.

DocHeader Item

Properties

The DocHeaders Collection consists of the following properties.

Count

Text

Count

Description

The number of items in the collection.

Syntax

object.**Count**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

Zero.

Range

>=zero.

Text

Description

Complete text of all headers in standard MIME header format.

Syntax

object.**Text** [= *String*]

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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.

Methods

The following section describes the methods performed by the DocHeaders Collection.

Add

Clear

Item

Remove

Add

Description

Adds a new item to the collection. The Name and Value parameters are converted to type String (STRING) 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*

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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

Description

Removes all items from the collection.

Return Value

Void.

Syntax

object.**Clear**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

Parameters

None.

Item

Description

Returns an 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*

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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

Description

Removes an item from the collection.

Return Value

Void.

Syntax

object.**Remove** *Index*

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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

A DocHeader object is an item in an DocHeaders collection. DocHeader items represent individual name and value pairs in MIME headers.

Name

Value

Name

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.**Item**("*Name*")

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

Permission

W(Read/Write).

Availability

R (Runtime).

Data Type

STRING.

Default Value

Empty string.

Range

N/A.

Value

Description

The item value which in MIME headers is the text after the label, colon character, and any leading spaces.

Syntax

object.**Item**("*Value*")

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

Permission

W(Read/Write).

Availability

R (Runtime).

Data Type

STRING.

Default Value

Empty string.

Range

N/A.

icErrors Collection

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.

Collection has its own properties and methods. It also has an item known as icErrors item.

Properties

Methods

icError Item

Properties

Following is a description of the properties in the icErrors collection.

Count

Source

Count

Description

The number of items in the collection.

Syntax

object.**Count**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

Zero.

Range

>=zero.

Source

Description

The vbObject that the most recent error applies to, or vbEmpty. Implementation is OCX-dependent. Unless specified by the specific control documentation, the value for Source is vbEmpty.

Syntax

object.**Source**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

Permission

R (Read only).

Availability

R (Runtime).

Data Type

VARIANT.

Default Value

None.

Range

N/A.

Methods

The following methods are performed by the icErrors Collection.

Clear

Item

Clear

Description

Removes all items from the collection and resets the Source property to vbEmpty.

Syntax

object.**Clear**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

Return Value

Void.

Parameters

None.

Item

Description

Returns an item from the collection.

Syntax

object.**Item** *Index*

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

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

A icError object is an item in an icErrors collection containing error messages. The properties supported by the icError item. are:

Code

Description

Type

Code

Description

Integer error code for the given error type.

Syntax

object.**Code**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Long.

Default Value

0.

Range

0-32767.

Description

Description

Text description of the error.

Syntax

object.**Description**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

Permission

R (Read only).

Availability

R (Runtime).

Data Type

STRING.

Default Value

Empty string.

Range

N/A.

Type

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**

The object placeholder is required and evaluates to the name of the relevant object or collection during execution.

Permission

R (Read only).

Availability

R (Runtime).

Data Type

STRING.

Default Value

Empty string.

Range

N/A.

Common Error Codes

The error numbers and messages in this category relate to more than one ActiveX control and are are in the range of 1000-2000.

Error Number	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
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.

ActiveX

ActiveX is a trademark for Microsoft Corporation