



Help for Whols/X

[Properties](#)

[Events](#)

[Methods](#)

[Interfaces](#)

How To Buy This Software

Order Form

Getting Custom Controls Written

Licensing Information

Description

The Mabry Whols/X control follows the Whols/NICNAME protocol and allows queries of InterNIC or other RFC 954 servers to obtain information about a user, domain or host. All domains and hosts registered in the InterNIC database are accessible, including the databases for North American domains, Government domains, Military domains, European domains and Asia-Pacific domains. Searches on partial strings or user handles can be completed, in addition to retrieving information on a specific domain, such as the registrant's company name, address, or contact information.

Whols/X comes as both a 32-bit ActiveX control and a 32-bit COM object, so you can use it in nearly any modern programming environment. -- ASP pages, Visual Basic applications, Visual C++ applications, anywhere that supports either COM objects or ActiveX controls. In addition to the normal event-driven programming model, Whols/X supports Fast Notifications. Fast Notifications allow your program to quickly receive events through simple functions, rather than going through the overhead to fire an event. This reduces the amount of time required to handle an event (of course, events are also supported). Further, Whols/X also supports non-blocking, pseudo-blocking and true blocking modes to meet almost any programming requirement.

ActiveX / OCX Object Name

Mabry.WholsX

ActiveX Compatibility

VB 4.0 (32-bit), 5.0 and 6.0

ActiveX Built With

Microsoft Visual C++ v6

ActiveX - Required DLLs

None. This is a light ActiveX control that requires no extra DLLs to operate. This means that your installation packages will be smaller. And, your web pages will load faster, since there are no extra DLLs to download.

Distribution Note When you develop and distribute an application that uses this control, you should install the control file into the user's Windows SYSTEM directory. The control file has version information built into it. So, during installation, you should ensure that you are not overwriting a newer version.

Close

Whols/X Properties

Properties that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

***Blocking** Property

***BlockingMode** Property

***Host** Property

***LibraryName** Property

***NotificationObject** Property

***Query** Property

***Response** Property

***ResponseCount** Property

***State** Property

***Timeout** Property

***Version** Property

Close

Whols/X Events

Events that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

*Done Event

*StateChanged Event

Close

WhoIs/X Methods

Methods that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

*GetWhoIs Method

Close

Whols/X Interfaces

Interfaces that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

*IWholsXComNotify Interface

*IWholsXCtlNotify Interface

Close

WhoIs/X IWhoIsXComNotify Interface Events

Events that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

*Done Event

*StateChanged Event

Close

WhoIs/X IWhoIsXCtlNotify Interface Events

Events that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

*Done Event

*StateChanged Event

How To Buy This Software

CREDITS

Whols/X was written by Zane Thomas.

CONTACT INFORMATION

Orders, inquiries, technical support, questions, comments, etc. can be sent to mabry@mabry.com on the Internet. Our mailing address/contact information is:

Mabry Software, Inc.
503 316th Street Northwest
Stanwood, WA 98292

Sales: 1-800-99-MABRY (U.S. Only)

Voice: 360-629-9278

Fax: 360-629-9278

Web: <http://www.mabry.com>

COST

The price of Whols/X (control only) is US\$25 (US\$30 for International orders). The cost of Whols/X and the C/C++ source code (of the control itself) is US\$75 (US\$80 for International orders).

Prices are subject to change without notice.

Printed manuals are available at US\$12.50 per copy.

PART NUMBERS

The product number for Whols/X (control only) is 18000.

The product number for and the C/C++ source code (of the control itself) is 18001.

DELIVERY METHODS

We can ship this software to you via air mail and/or e-mail.

Air Mail - you will receive diskettes, a printed manual (if purchased), and printed receipt if you choose this delivery method. The costs are:

US\$10.00	US Priority Mail
US\$15.00	Airborne Express 2nd Day (US deliveries only)
US\$20.00	Airborne Express Overnight (US deliveries only)
US\$20.00	Global Priority Mail (Int'l deliveries only; Western Europe, Pacific Rim and Canada only)
US\$45.00	International Airborne Express (Int'l deliveries only)

E-Mail - We can ship this package to you via e-mail. You need to have an e-mail account that can accept large file attachments (which includes CompuServe, AOL, and most Internet providers). We will e-mail a receipt to you.

Be sure to include your full mailing address with your order. Sometimes (on the Internet) the package cannot be e-mailed, so we are forced to send it through the normal mails.

ORDER / PAYMENT METHODS

You can order this software by phone, fax, e-mail, mail. For your convenience, an order form has been provided that you can print out directly from this help file.

Please note that orders must include all information that is requested on our order form. Your shipment WILL BE DELAYED if we have to contact you for additional information (such as phone number, street address, etc.).

You can pay by credit card (VISA, MasterCard, American Express, Discover, NOVUS), check (U.S. dollars drawn on a U.S. bank), cash, International Money Order, International Postal Order, Purchase Order (established business entities only - terms net 30), or wire transfer.

WIRE TRANSFER INFORMATION

Here is the information you need regarding our account for a wire funds transfer:

Bank Name:	SeaFirst - Stone Way Branch
Bank Address:	3601 Stone Way North Seattle, WA 98103
Bank Phone:	206-585-4951
Account Name:	Mabry Software, Inc.
Routing Number:	12000024
Account Number:	16311706

If you are paying with a wire transfer of funds, please add US\$25.00 to your order. This is the fee that SeaFirst Bank charges Mabry Software. Also, please ADD ANY ADDITIONAL FEES THAT YOUR BANK MAY CHARGE for wire transfer service. If you are paying with a wire transfer, we must have full payment deposited to our account before we can ship your order.

Copyright © 1998 by Mabry Software, Inc.



Whols/X Order Form

Use the Print Topic... command from the File menu to print this order form.

Mail this form to: Mabry Software, Inc.
503 316th Street Northwest
Stanwood, WA 98292

Phone: 360-629-9278
Fax: 360-629-9278
Internet: mabry@mabry.com
Web: www.mabry.com

Where did you get this copy of Whols/X?

Name: _____

Ship to: _____

Phone: _____

Fax: _____

E-Mail: _____

Credit Card #: _____ exp. _____

P.O. # (if any): _____ Signature _____

qty ordered _____ REGISTRATION
\$25.00 (\$30.00 international). Check or money order in U.S. currency drawn on a U.S. bank. Add \$10.00 per order for shipping and handling. Add \$12.50 per printed manual.

qty ordered _____ SOURCE CODE AND REGISTRATION
\$75.00 (\$80.00 international). Check or money order in U.S. currency drawn on a U.S. bank. Add \$10.00 per order for shipping and handling. Add \$12.50 per printed manual.

Error Codes

Constant	Value	Description
	0	No error.
WSAEINTR	10004	System level interrupt interrupted socket operation.
WSAEBADF	10009	Generic error for invalid format, bad format.
WSAEACCES	10013	Generic error for access violation.
WSAEFAULT	10014	Generic error for fault.
WSAEINVAL	10022	Generic error for invalid format, entry, etc.
WSAEMFILE	10024	Generic error for file error.
	10025	The IP address provided is not valid or the host specified by the IP does not exist.
WSAENOTSOCK	10038	Invalid socket or not connected to remote.
WSAEADDRINUSE	10048	The specified address is already in use.
WSAEADDRNOTAVAIL	10049	The specified address is not available.
WSAENETDOWN	10050	The connected network is not available.
WSAENETUNREACH	10051	The connected network is not reachable.
WSAENETRESET	10052	The connected network connection has been reset.
WSAECONNABORTED	10053	The current connection has been aborted by the network or intermediate services.
WSAECONNRESET	10054	The current socket connection has been reset.
WSAENOTCONN	10057	The current socket has not been connected.
WSAESHUTDOWN	10058	The connection has been shutdown.
WSAETIMEDOUT	10060	The current connection has timed out.
WSAECONNREFUSED	10061	The requested connection has been refused by the remote host.
WSAENAMETOOLONG	10063	Specified host name is too long.
WSAEHOSTDOWN	10064	Remote host is currently unavailable.
WSAEHOSTUNREACH	10065	Remote host is currently unreachable.
WSASYSNOTREADY	10091	Remote system is not ready.
WSAVERNOTSUPPORTED	10092	Current socket version not supported by application.
WSANOTINITIALISED	10093	Socket API is not initialized.
WSAEDISCON	10101	Socket has been disconnected.

See Also

[**BlockingMode** Property](#)

[**Done** Event](#)

[**GetWhols** Method](#)

[**Host** Property](#)

[**Query** Property](#)

[**Timeout** Property](#)

See Also

[**Blocking** Property](#)

[**GetWhoIs** Method](#)

[**Host** Property](#)

[**Query** Property](#)

[**Timeout** Property](#)

See Also

[**Blocking** Property](#)

[**GetWhoIs** Method](#)

[**Host** Property](#)

[**Query** Property](#)

[**Response** Property](#)

[**ResponseCount** Property](#)

See Also

[**Blocking** Property](#)

[**BlockingMode** Property](#)

[**Done** Event](#)

[**Host** Property](#)

[**IWholsXComNotify** Done Event](#)

[**IWholsXComNotify** StateChanged Event](#)

[**IWholsXCtlNotify** Done Event](#)

[**IWholsXCtlNotify** StateChanged Event](#)

[**LibraryName** Property](#)

[**Query** Property](#)

[**Response** Property](#)

[**ResponseCount** Property](#)

[**State** Property](#)

[**StateChanged** Event](#)

[**Timeout** Property](#)

See Also

[**Blocking** Property](#)

[**BlockingMode** Property](#)

[**Done** Event](#)

[**GetWhols** Method](#)

[**Timeout** Property](#)

See Also

[IWholsXComNotify Done Event](#)

[IWholsXComNotify StateChanged Event](#)

[NotificationObject Property](#)

See Also

[IWholsXCtrlNotify Done Event](#)

[IWholsXCtrlNotify StateChanged Event](#)

[NotificationObject Property](#)

See Also

[GetWhols Method](#)

See Also

[IWholsXComNotify Done Event](#)

[IWholsXComNotify Interface](#)

[IWholsXComNotify StateChanged Event](#)

[IWholsXCtlNotify Done Event](#)

[IWholsXCtlNotify Interface](#)

[IWholsXCtlNotify StateChanged Event](#)

See Also

[**Blocking** Property](#)

[**BlockingMode** Property](#)

[**Done** Event](#)

[**GetWhols** Method](#)

[**Response** Property](#)

See Also

[Done Event](#)

[GetWhoIs Method](#)

[Query Property](#)

[ResponseCount Property](#)

See Also

[Done Event](#)

[GetWhoIs Method](#)

[Response Property](#)

See Also

[GetWhols Method](#)

[IWholsXComNotify StateChanged Event](#)

[IWholsXCtlNotify StateChanged Event](#)

[StateChanged Event](#)

See Also

[GetWhols Method](#)

[State Property](#)

See Also

[**Blocking** Property](#)

[**BlockingMode** Property](#)

[**GetWhols** Method](#)

[**Host** Property](#)

See Also

[GetWhols](#) Method

[IWholsXComNotify](#) Interface

[NotificationObject](#) Property

See Also

[**GetWhols** Method](#)

[**IWholsXComNotify** Interface](#)

[**NotificationObject** Property](#)

[**State** Property](#)

See Also

[GetWhols](#) Method

[IWholsXctlNotify](#) Interface

[NotificationObject](#) Property

See Also

[**GetWhols** Method](#)

[**IWholsXctlNotify** Interface](#)

[**NotificationObject** Property](#)

[**State** Property](#)

BlockingMode Property

[See Also](#)

Description

Determines message processing when [Blocking](#) is True.

Syntax

object.**BlockingMode** [= *BlockingMode*]

The syntax of the **BlockingMode** property has these parts:

<u>Part</u>	<u>Description</u>
<i>object</i>	A Whols/X control.
<i>BlockingMode</i>	Integer that determines whether Blocking is true blocking or pseudo blocking.

Remarks

There are two different block modes to choose from, `soxTrueBlocking` (0) and `soxPseudoBlocking` (1). If you have Blocking enabled and `soxTrueBlocking` mode selected, then blocking calls will process only `WM_PAINT` messages for your program. All other messages will be ignored.

If you choose `soxPseudoBlocking` mode then all of your program's messages will be processed, including mouse, keyboard, and so on. PseudoBlocking allows full UI interactivity while blocking operations are in progress, but also requires that you exercise considerable care to ensure that you do not call any other blocking functions while one is in progress. If you fail to handle this correctly, errors will result (which you may choose to handle with `On Error Goto` or similar statements).

Data Type

`BlockingModesEnum`

Blocking Property

[See Also](#)

[Error Codes](#)

Description

Determines if methods are blocking (synchronous) or non-blocking (asynchronous).

Syntax

object.**Blocking** [= *boolean*]

The syntax of the **Blocking** property has these parts:

<u>Part</u>	<u>Description</u>
<i>object</i>	A Whols/X control.
<i>boolean</i>	A boolean expression that determines if the control waits for completion of an operation (True), or returns control to the program immediately (False).

Remarks

If this property is set to True, the [GetWhols](#) method will not return to your code until the command completes. In other words, the command will be handled synchronously and your code will execute sequentially. The [BlockingMode](#) property controls the type of events that will be processed while waiting for blocking calls to return.

If this property is false, the GetWhols action is handled asynchronously. The execution of the action will be event driven and the next line of code will execute immediately. The [Done event](#) will fire to notify your application when the action has completed. Windows is inherently event driven and non-blocking mode is recommended so that your program can respond to user actions while the GetWhols action is in progress.

Important Note: The Blocking property must not be changed while a connection is established. You should only change the Blocking property when the control is not connected to a server. Changing the Blocking property while connected could cause some strange behavior in the control.

Data Type

Boolean

Done Event

[See Also](#)

[Error Codes](#)

Description

This event procedure is fired when the [GetWhoIs](#) query is completed.

Syntax

Sub *object_Done* (*[index As Integer,* *LastMethod As MethodsEnum,* *ErrorCode As Integer]*)

The syntax of the **Done** event has these parts:

Part	Description
<i>object</i>	A WhoIs/X control.
<i>index</i>	An integer that identifies a control if it's in a control array.
<i>LastMethod</i>	Contains the last method executed. With WhoIs/X, this will always be 0 (GetWhoIsMethod)
<i>ErrorCode</i>	An integer containing the error code, if any.

Remarks

This event fires when the data request completes. If the ErrorNumber parameter is zero, the operation completed correctly. If ErrorNumber is not zero, then an error occurred.

IWhoIsXCtlNotify Interface Done Event

[See Also](#)

[Error Codes](#)

Description

This event procedure is fired when the control completes the [GetWhoIs method](#) and the [IWhoIsXCtlNotify](#) interface has been implemented.

Syntax

Sub *object_Done* (*[index As Integer,* *WhoIsCtl As ByRef IWhoIsXCtl, LastMethod As MethodsEnum, ErrorCode As Integer]*)

The syntax of the **Done** event has these parts:

<u>Part</u>	<u>Description</u>
<i>object</i>	Required. An instance of an IWhoIsXCtlNotify interface.
<i>WhoIsCtl</i>	A WhoIs/X control.
<i>LastMethod</i>	Contains the last method executed. With WhoIs/X, this will always be 0 (GetWhoIsMethod)
<i>ErrorCode</i>	An integer containing the error code, if any.

Remarks

This event fires when the data request completes. If the ErrorNumber parameter is zero, the operation completed correctly. If ErrorNumber is not zero, then an error occurred.

IWholsXComNotify Interface Done Event

[See Also](#)

[Error Codes](#)

Description

This event procedure is fired when the COM object completes the [GetWhols method](#) and the [IWholsXComNotify interface](#) has been implemented.

Syntax

Sub *object_Done* (*[index As Integer,]* *WholsX As ByRef WhoisXCom*, *LastMethod As MethodsEnum*, *ErrorCode As Integer*)

The syntax of the **Done** event has these parts:

<u>Part</u>	<u>Description</u>
<i>object</i>	Required. An instance of an IWholsXComNotify interface.
<i>WholsX</i>	A Whols/X COM Object.
<i>LastMethod</i>	Contains the last method executed. With Whols/X, this will always be 0 (GetWhoisMethod)
<i>ErrorCode</i>	An integer containing the error code, if any.

Remarks

This event fires when the data request completes. If the ErrorNumber parameter is zero, the operation completed correctly. If ErrorNumber is not zero, then an error occurred.

GetWhols Method

[See Also](#)

[Error Codes](#)

Description

Gets nickname / Whols information from specified server.

Syntax

object.**GetWhols***Host, Query*

The syntax of the **GetWhols** method has these parts:

Part	Description
<i>object</i>	Required. A Whols/X control.
<i>Host</i>	Optional. Optional string expression indicating the host name or IP of the server.
<i>Query</i>	Optional. Optional string expression containing the query to send to the server.

Remarks

This method requests the nickname / Whols information specified by the *query* using the *host*. If no host is specified, the Host property is used. If no query is specified, the Query property is used.

This method can be called in one of two ways: blocking or non-blocking as determined by the Blocking property.

If blocking = True, the Whols/X control does not return control to your program until it retrieves the queried data from the host specified or until it times-out. BlockingMode determines whether your applicatoin will process any user interface messages while waiting for the action to complete.

With non-blocking mode (recommended) the control returns immediately. The control fires the Done event upon completion (good or bad).

Sometimes error 10035 may be reported. This is only a warning from the winsock layer and can be treated as normal (ignored).

Host Property

[See Also](#)

Description

IP address or host name of the WhoIs server.

Syntax

object.**Host** [= *Host*]

The syntax of the **Host** property has these parts:

Part	Description
<i>object</i>	A WhoIs/X control.
<i>Host</i>	A string expression that specifies the name or IP address of a WhoIs server.

Remarks

This property determines which server the WhoIs control queries to find nickname / WhoIs information. Specify the host name (or IP address directly) of the server to use during the [GetWhoIs](#) method.

Typical settings would be:

whois.internic.net	North American domains
whois.nic.mil	Military domains
whois.nic.gov	Government domains
whois.arin.net	American Registry domains
whois.ripe.net	European domains
whois.apnic.net	Asia Pacific domains

Data Type

String

IWhoIsXComNotify Interface

[See Also](#)

[Events](#)

[Error Codes](#)

Description

Fast Notification interface for the COM Object (DLL).

Remarks

The WhoIs/X COM Object implements a Fast Notification (early-bound callback) interface as an alternative to using events. You can implement the early-bound callback functions which are identical to the events except that an instance of the COM Object calling the function is passed as the first parameter. This extra parameter substitutes for the Index parameter you would have had in an array implementation.

To implement the IWhoIsXComNotify interface, add the following in the declarations section of a form or class:

```
Dim WithEvents WhoIsX As WhoIsXCom  
Implements IWhoIsXComNotify
```

Then, use the [NotificationObject property](#) to specify the object which will receive the notifications, such as:

```
Private Sub Form_Load ()  
    Set WhoIsX = New WhoIsXCom  
    WhoIsX.NotificationObject = Me  
End Sub
```

Finally, use the [IWhoIsXComNotify Done event](#) and the [IWhoIsXComNotify StateChanged event](#) to capture the events.

IWhoIsXCtlNotify Interface

[See Also](#)

[Events](#)

[Error Codes](#)

Description

Fast Notification interface for the OCX control.

Remarks

The WhoIs/X OCX implements a Fast Notification (early-bound callback) interface as an alternative to using events. You can implement the early-bound callback functions which are identical to the events except that an instance of the OCX calling the function is passed as the first parameter. This extra parameter substitutes for the Index parameter you would have had in a control array implementation.

To implement the IWhoIsXCtlNotify interface, add the following in the declarations section of a form or class:

```
Implements IWhoIsXCtlNotify
```

Then, use the [NotificationObject property](#) to specify the object which will receive the notifications, such as:

```
Private Sub Form_Load ()  
    WhoIsXCtl1.NotificationObject = Me  
End Sub
```

Finally, use the [IWhoIsXCtlNotify Done event](#) and the [IWhoIsXCtlNotify StateChanged event](#) to capture the events.

LibraryName Property

[See Also](#)

[Error Codes](#)

Description

Specifies the name of the socket stack (winsock) DLL.

Syntax

object.**LibraryName** [= *LibraryName*]

The syntax of the **LibraryName** property has these parts:

<u>Part</u>	<u>Description</u>
<i>object</i>	A WhoIs/X control.
<i>LibraryName</i>	A string expression that specifies the winsock DLL to use.

Remarks

The default value for this property is "wsock32.dll", which is the socket stack that Microsoft ships with 32-bit systems. If you have a different socket stack provider, you can use the provider's socket stack by specifying the provider's DLL name.

Data Type

String

NotificationObject Property

[See Also](#)

[Error Codes](#)

Description

Specifies the object that has implemented an interface to receive the Fast Notifications.

Syntax

object.**NotificationObject** [= *NotificationObject*]

The syntax of the **NotificationObject** property has these parts:

Part	Description
<i>object</i>	A Whols/X control.
<i>NotificationObject</i>	IWhoisXCtlNotify when implementing the IWhoisXCtlNotify interface and IWhoisXComNotify when implementing the IWhoisXComNotify interface

Remarks

Both the Whols/X COM object and control support fast-notification callback interfaces. If you implement [IWhoisXCtlNotify](#) or [IWhoisXComNotify](#) in a form or class, you can assign an instance of that form or class to the Whols/X object or control's NotificationObject. This will allow the form or class to receive event notifications directly.

For example, to implement Fast Notifications for the Whols/X control for a form, implement the interface in the Declarations section with:

```
Implements IWhoisXCtlNotify
```

Then, to have the form receive the notifications, do:

```
Private Sub Form_Load ()  
    WhoisXCtl1.NotificationObject = Me  
End Sub
```

Data Type

IWhoisXCtlNotify

Query Property

[See Also](#)

Description

This holds the string that Whols/X uses to query the nickname / Whols server.

Syntax

object.**Query** [= *Query*]

The syntax of the **Query** property has these parts:

Part	Description
<i>object</i>	A Whols/X control.
<i>Query</i>	A string expression contains the Whols query.

Remarks

This property holds the string that you are looking for in the nickname / Whols database. There are several ways to specify your query.

First, you can simply specify a name, such as "mabry" or "mabry.com". Searching this way will usually return a list of matches. If your search yields more than one match, some of the lines returned in [Response](#) will look like:

```
mabry.com (MABRY-DOM) ...  
james@mabry.com (JS716) ...
```

The text in parentheses is the handle for that site or user. You can specify a handle in the Query property to find that specific entry by prefacing the query with the "!" character, such as:

```
!MABRY-DOM
```

If you are looking for an address, you can use the @ symbol to search for users. For example: "@mabry" will find everyone at "mabry". "james@mabry" will find the user named "james" at "mabry". "james@" will find all users named "james".

Add a trailing '.' to your query to do a partial search that matches everything *starting* with your input. Example, "mack." finds names "Mack", "Mackay", etc.

Add a leading '.' to your query to do a partial search that matches everything *ending* with your input. Example, ".mack" finds names "Mack", "Romack", etc.

Be careful when using "@" and "." searches as they can retrieve huge lists.

Data Type

String

ResponseCount Property

[See Also](#)

[Error Codes](#)

Description

Indicates the number of elements in the [Reponse](#) array (lines of data retrieved).

Syntax

object.**ResponseCount** [= *ResponseCount*]

The syntax of the **ResponseCount** property has these parts:

<u>Part</u>	<u>Description</u>
<i>object</i>	A Whols/X control.
<i>ResponseCount</i>	An integer that holds the number of elements in the Response property array.

Remarks

ResponseCount holds the number of lines of data received from the search. This property tells you how many lines of text are found in the [Response](#) array. This property is only valid after searching.

This property is read-only and only available at run-time.

Data Type

Integer

Response Property

[See Also](#)

[Error Codes](#)

Description

Response is a string array that holds the data retrieved from a query.

Syntax

object.**Response**(*index*) [= *Response*]

The syntax of the **Response** property has these parts:

<u>Part</u>	<u>Description</u>
<i>object</i>	A Whois/X control.
<i>index</i>	A long integer that identifies an element in the response array.
<i>Response</i>	A string that contains a response value.

Remarks

This property contains the data retrieved from the nickname / Whois database. It is a string array where each element in the array contains a line of text in order, as received from the Whois server (database). The index ranges from zero (0) to ([ResponseCount](#) - 1).

The data received can be in various formats (freeform at times).

This property is only valid after searching using the [GetWhois method](#).

This property is read-only and only available at run-time.

Data Type

String

State Property

[See Also](#)

[Error Codes](#)

Description

Returns the current state of the control.

Syntax

object.State [= *State*]

The syntax of the **State** property has these parts:

<u>Part</u>	<u>Description</u>
<i>object</i>	A Whols/X control.
<i>State</i>	Integer that indicates the current state of the socket.

Remarks

This property may take on many values while executing commands. No additional methods may be executed while State is greater than 1 (Created).

This property is read-only and only available at run-time.

The following constants represent the set of values that are assigned to the State property:

<u>Constant</u>	<u>Value</u>	<u>Description</u>
Unusable	0	The socket is unusable. This occurs when the socket stack specified by LibraryName cannot be loaded.
Closed	1	The socket is usable, but not yet created.
Created	2	The socket has been created.
Connecting	3	The socket is in the process of connecting to the server.
Connected	4	The socket has been connected. State is set to Connected when a connection is established with the server.
Waiting	5	The socket is waiting for response from the server.
Retrieving	6	The socket is retrieving the data from the server.
Done	7	The socket has finished the query and closed the connection.

Data Type

StatesEnum

StateChanged Event

[See Also](#)

[Error Codes](#)

Description

This event procedure is fired as the GetWhols query progresses.

Syntax

Sub *object* **StateChanged**(*[index As Integer,* *OldState As StatesEnum,* *NewState As StatesEnum)*

The syntax of the **StateChanged** event has these parts:

<u>Part</u>	<u>Description</u>
<i>object</i>	A Whols/X control.
<i>index</i>	An integer that identifies a control if it's in a control array.
<i>OldState</i>	Integer indicating the previous state value.
<i>NewState</i>	Integer indicating the current state value.

Remarks

As the nickname / Whols request progresses, the StateChanged event fires to indicate the current state of the transaction.

The *OldState* and *NewState* parameters contain the previous and current states. See the [State property](#) for a list of the possible conditions.

IWholsXComNotify Interface StateChanged Event

[See Also](#)

[Error Codes](#)

Description

This event procedure is fired as the [GetWhols method](#) progresses *if* the [IWholsXComNotify interface](#) has been implemented.

Syntax

Sub *object_StateChanged* (*[index As Integer,]* *WholsX As ByRef WhoisX*, *OldState As StatesEnum*, *NewState As StatesEnum*)

The syntax of the **StateChanged** event has these parts:

<u>Part</u>	<u>Description</u>
<i>object</i>	Required. An instance of an IWholsXComNotify interface.
<i>WholsX</i>	A Whols/X COM Object.
<i>OldState</i>	Integer indicating the previous state value.
<i>NewState</i>	Integer indicating the current state value.

Remarks

As the nickname / Whols request progresses, the StateChanged event fires to indicate the current state of the transaction.

The *OldState* and *NewState* parameters contain the previous and current states. See the [State property](#) for a list of the possible conditions.

IWhoIsXCtlNotify Interface StateChanged Event

[See Also](#)

[Error Codes](#)

Description

This event procedure is fired as the [GetWhoIs method](#) progresses *if* the [IWhoIsXCtlNotify interface](#) has been implemented.

Syntax

Sub *object_* **StateChanged**(*[index As Integer,]* *WhoIsCtl As ByRef IWhoIsXCtl*, *OldState As StatesEnum*, *NewState As StatesEnum*)

The syntax of the **StateChanged** event has these parts:

<u>Part</u>	<u>Description</u>
<i>object</i>	Required. An instance of an IWhoIsXCtlNotify interface.
<i>WhoIsCtl</i>	A WhoIs/X control.
<i>OldState</i>	Integer indicating the previous state value.
<i>NewState</i>	Integer indicating the current state value.

Remarks

As the nickname / WhoIs request progresses, the StateChanged event fires to indicate the current state of the transaction.

The *OldState* and *NewState* parameters contain the previous and current states. See the [State property](#) for a list of the possible conditions.

Timeout Property

[See Also](#)

[Error Codes](#)

Description

The amount of time (in seconds) the control will wait for an operation..

Syntax

object.Timeout [= *Timeout*]

The syntax of the **Timeout** property has these parts:

<u>Part</u>	<u>Description</u>
<i>object</i>	A Whols/X control.
<i>Timeout</i>	A long integer that specifies the amount of time to wait, in seconds.

Remarks

This property determines how long the control will wait for an action to complete before declaring an error. This property is measured in seconds. If the server fails to respond within this time period, an error occur.

Data Type

Long

Version Property

[Error Codes](#)

Description

Returns the version of the control.

Syntax

object.**Version** [= *Version*]

The syntax of the **Version** property has these parts:

<u>Part</u>	<u>Description</u>
<i>object</i>	A Whols/X control.
<i>Version</i>	String containing the version number of the control.

Remarks

This property holds the current version of the control. It is read-only and available at both design-time and run-time.

Data Type

String

Getting Custom Controls Written

If you or your organization would like to have custom controls written, you can contact us at the following:

Mabry Software, Inc.
503 316th Street Northwest
Stanwood, WA 98292
Phone: 360-629-9278
Fax: 360-629-9278
Internet: mabry@mabry.com

You can also contact Zane Thomas. He can be reached at:

Zane Thomas
Post Office Box 121
Indianola, WA 98342
Internet: zane@mabry.com

Licensing Information

Legalese Version

Mabry Software grants a license to use the enclosed software to the original purchaser. Copies may be made for back-up purposes only. Copies made for any other purpose are expressly prohibited, and adherence to this requirement is the sole responsibility of the purchaser.

Customer written executable applications containing embedded Mabry products may be freely distributed, without royalty payments to Mabry Software, provided that such distributed Mabry product is bound into these applications in such a way so as to prohibit separate use in design mode, and that such Mabry product is distributed only in conjunction with the customers own software product. The Mabry Software product may not be distributed by itself in any form.

Neither source code for Mabry Software products nor modified source code for Mabry Software products may be distributed under any circumstances, nor may you distribute .OBJ, .LIB, etc. files that contain our routines. This control may be used as a constituent control only if the compound control thus created is distributed with and as an integral part of an application. Permission to use this control as a constituent control does not grant a right to distribute the license (LIC) file or any other file other than the control executable itself. This license may be transferred to a third party only if all existing copies of the software and its documentation are also transferred.

This product is licensed for use by only one developer at a time. Mabry Software expressly prohibits installing this product on more than one computer if there is any chance that both copies will be used simultaneously. This restriction also extends to installation on a network server, if more than one workstation will be accessing the product. All developers working on a project which includes a Mabry Software product, even though not working directly with the Mabry product, are required to purchase a license for that Mabry product.

This software is provided as is. Mabry Software makes no warranty, expressed or implied, with regard to the software. All implied warranties, including the warranties of merchantability and fitness for a particular use, are hereby excluded.

MABRY SOFTWARE'S LIABILITY IS LIMITED TO THE PURCHASE PRICE. Under no circumstances shall Mabry Software or the authors of this product be liable for any incidental or consequential damages, nor for any damages in excess of the original purchase price.

To be eligible for free technical support by telephone, the Internet, CompuServe, etc. and to ensure that you are notified of any future updates, please complete the enclosed registration card and return it to Mabry Software.

English Version

We require that you purchase one copy of a control per developer on a project. If this is met, you may distribute the control with your application royalty free. You may never distribute the LIC file. You may not change the product in any way that removes or changes the requirement of a license file.

We encourage the use of our controls as constituent controls when the compound controls you create are an integral part of your application. But we don't allow distribution of our controls as constituents of other controls when the compound control is not part of an application. The reason we need to have this restriction is that without it someone might decide to use our control as a constituent, add some trivial (or even non-trivial) enhancements and then sell the compound control. Obviously there would be little difference between that and just plain reselling our control.

If you have purchased the source code, you may not re-distribute the source code either (nor may you copy it into your own project). Mabry Software retains the copyright to the source code.

Your license is transferable. The original purchaser of the product must make the transfer request. Contact us for further information.

The sample versions of our products are intended for evaluation purposes only. You may not use the sample version to develop completed applications.

