Help for ASocket

Properties Events Methods Frequently Asked Questions

Registration Information

Order Form

Getting Custom Controls Written

Licensing Information

Description

ASocket VBX/OCX is a Visual Basic custom control that provides you with full access to the power of Windows, making it easy to write TCP/IP client and server software. ASocket provides complete support for Windows and Visual Basic's event-driven programming model and lets you do almost anything that can be done with WinSock.

SocketType sockStream provides sequenced, reliable, full-duplex, connection-based byte streams. Uses the Transmission Control Protocol (TCP) for the Internet address family. SocketType sockDatagram supports datagrams, which are connectionless, unreliable packets (typically small) of a fixed maximum length. Uses the User Datagram Protocol (UDP) for the Internet address family.

File Name

ASOCKET1.VBX, ASOCK32.OCX

ActiveX / OCX Object Name

Mabry.AsyncSocketControl

ActiveX Compatility

VB 4.0 (32-bit) and 5.0

ActiveX Built With

Microsoft Visual C++ v4

ActiveX - Required DLLs

MFC40.DLL (October 6th, 1995 or later)
OLEPRO32.DLL (October 6th, 1995 or later)
MSVCRT40.DLL (September 29th, 1995 or later)

VBX Object Type

MabryAsyncSocket

VBX Compatibility

VB 2.0, 3.0 and 4.0 (16-bit)

VBX Built With

Microsoft Visual C++ v1.5

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

ASocket Properties

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

Close

ASocket Events

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

*__OnAccept__ Event

*__OnClose__ Event

*__OnConnect__ Event

*__OnOutOfBandData__ Event

*__OnReceive__ Event

*__OnSend__ Event

Close

ASocket Methods

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

*__Bind__ Method
*__Close__ Method
*__Connect__ Method
*__Create__ Method
*__Listen__ Method
*__Receive__ Method
*__ReceiveFrom__ Method
*__Send__ Method
*__SendTo__ Method

Close

Frequently Asked Questions

General Questions

- How do I get the local machine's IP address without doing a connect?
- What is a socket? How does it work? How does that relate to Mabry's ASocket control?
- I want a write a chat room program. Is it possible to allow one Mabry ASocket control to handle multiple connections?
- What port number should I use? I read in some documentation somewhere that any port number over 1000 would be good, but could you please give me a min-max range?
- Why am I getting errors when sending large amounts of data (either in big chunks, or in many small chunks)?
- Can I create a sniffer with the ASocket control? I want to capture every packet on a small ethernet.
- How can I shorten the Timeout period during a Connect? My program waits for 2 minutes (in the case of no connection) before returning.
- I want to make sure that all of my data has been sent before closing the socket and unloading the control. Since there is no SendComplete event, how do I do this?
- Is it possible for the control to fire an event after the socket has been closed?
- Why do I get "Generic Error For Invalid Format, Entry, etc." when I try to establish a connection?
- Why am I able to connect using the RemoteAddress property, but not when I use the RemoteName property?
- What is the difference between a Datagram socket and a Stream socket?

Internet Pack - General Questions

- Why won't my Internet Pack VBXes load into VB?
- Why do I get a GPF when I try to unload my form (or control) from the Done event?
- With which TCP/IP stacks have your Internet controls been tested?
- How do I enable/disable the Windows 95 Dial-Up Networking connect prompt when my application issues a Connect method?
- Why won't my Internet Pack VBX load?
- How do I convert my code from BLOCKING (Synchronous) to NON-BLOCKING (Asynchronous)?
- How can I detect whether someone has entered an IP or host name?
- What is the meaning of Error 20002 "unexpected server response"?
- Can you recommend any good books that will help me understand Internet programming better?

Internet Pack - Blocking

- I'm unclear on blocking. Can you explain it to me?
- Should I use blocking or non-blocking calls?
- Why do I keep getting the error "Busy executing asynchronous command"?
- Why do I keep getting errors when using an Internet VBX control?

Internet Pack - Debugging

- How do I tell what's happening when your control is talking to a server?

How do I get the local machine's IP address without doing a connect?

Frequently Asked Questions

Many people complain about having to connect to some remote socket before the local machine's address can be obtained. This is an unfortunate "feature" of WinSock. But, there is a work around (this applies to the 32-bit ASocket OCX only).

You still need an Internet connection for this method to work. You just don't have to explicitly connect to a port in order to get the local IP address.

Here is the code to retrieve the local machine's IP address:

```
Private Declare Function GetComputerName Lib "kernel32" Alias
"GetComputerNameA" (ByVal lpBuffer As String, nSize As Long) As Long
Private Const MAX_COMPUTERNAME_LENGTH As Long = 15&

Public Function CurrentMachineName() As String
    Dim lSize As Long
    Dim sBuffer As String

    sBuffer = Space(MAX_COMPUTERNAME_LENGTH + 1)
    lSize = Len(sBuffer)

    If GetComputerName(sBuffer, lSize) Then
        CurrentMachineName = Left$(sBuffer, lSize)
    End If
End Function

Private Sub Form_Load()
    Dim name As String

    Me.Show
    Print "Machine Name: ";
    name = CurrentMachineName()
    If (name = "") Then
        Print "unknown"
    Else
        Print name
        MSock1.RemoteNameAddrXlate = True
        MSock1.RemoteName = LCase$(name)
        Print "Address: " & MSock1.RemoteAddress
    End If
End Sub
```

What is a socket? How does it work? How does that relate to Mabry's ASocket control?

Frequently Asked Questions

A socket simply consists of an IP address, the program's port number (just don't use a well-known port number, usually something over 1000 is safe), and whether it's TCP (guaranteed end-to-end integrity) or UDP protocol (not guaranteed end-to-end integrity).

In a nutshell, you need to open the connection(s), send/receive data, and close the connection(s).

To open:
1. Specify the remote IP address (RemoteAddress property)
2. Specify the remote port number (RemotePort property)
3. Specify the local IP address (LocalAddress property)
4. Specify the local port number (LocalPort property)
5. Use the Create method
6. Use the Connect method

To send data to the remote system, load up the SendBuffer property with the desired data and issue the Send method.

To receive data, use the OnReceive event to capture the incoming data.

To close, simply use the Close method.

I want a write a chat room program. Is it possible to allow one Mabry ASocket control to handle multiple connections?

This can be done by using datagrams. The "server" would receive what amounts to connect-to-chat requests from clients and then maintain them in a list. When data arrives from one client it would be retransmitted to all other clients using the SendTo method.

The alternative is to use a connection-oriented scheme (stream instead of datagram) and use multiple controls.

What port number should I use? I read in some documentation somewhere that any port number over 1000 would be good, but could you please give me a min-max range?

Frequently Asked Questions

0 - 65535. Although, 0 through 1023 are reserved for known services, so don't write any custom services using those ports (although you are certainly free to create custom clients to access services on well known ports).

Why am I getting errors when sending large amounts of data (either in big chunks, or in many small chunks)?

The OnSend event is fired whenever the socket control becomes ready to send data after some period of time when it was not ready to send data. There are two situations where a socket makes this transition. When a socket is connected it becomes ready to send and the OnSend event is fired. If data is being sent to through the socket and the output queue becomes full the socket becomes not-ready to send data and either a 10035 or 20000 error is thrown. After the control once again becomes ready for data the OnSend event is fired and the program can resume sending where it left off.

The 10035 error is thrown when a Send is requested and NONE of the bytes are queued for transmission

The 20000 error is thrown when a Send is requested and some of the bytes in the SendBuffer have been queued for transmission. In this case the BytesSent property will contain the number of bytes which were queued. That number of bytes should be removed from the SendBuffer property before the next Send method is executed (this Send will most logically occur when the OnSend event is fired).

Can I create a sniffer with the ASocket control? I want to capture every packet on a small ethernet.

Frequently Asked Questions

No, you can't create a sniffer with the Asocket control because the Asocket control gives you direct access to the Winsock interface which is operating at the Transport layer, not the Network layer. Thus, you only have access to packets that are destined for your socket [IP, Port, service (TCP or UDP)).

How can I shorten the Timeout period during a Connect? My program waits for 2 minutes (in the case of no connection) before returning.

Frequently Asked Questions

The Timeout property is only to be used after connecting to the server (to handle server timeouts vs. WinSock timeouts). WinSock is controlling the connect timeout.

To abort a connect and unload your application if a connection doesn't occur within 15 seconds, set Blocking to False and use a standard Timer control set to 15 seconds to unload the form when it fires. Note -- don't forget to disable the timer if the connect is successful!

I want to make sure that all of my data has been sent before closing the socket and unloading the control. Since there is no SendComplete event, how do I do this?

Frequently Asked Questions

Set LingerEnabled = True, LingerMode = 1 and LingerTime as desired. When you invoke the Close method, the call will not return until the data has been sent.

Is it possible for the control to fire an event after the socket has been closed?

Frequently Asked Questions

A WM_SOCKET (connect, data receive, etc.) message may already be wandering through the system when Close is called. When it arrives at the control it will generate an event. So it is possible for the control to fire an event just after a Close.

Why do I get "Generic Error For Invalid Format, Entry, etc." when I try to establish a connection?

Frequently Asked Questions

This can happen if an address is not in the w.y.y.z format. If the IP is not known and you are using RemoteName and RemoteNameAddrXlate to connect, set the RemoteAddress to 0.0.0.0. If using the ReceieveFrom method, set the RemoteAddress to 0.0.0.0 prior to receiving datagrams.

Why am I able to connect using the RemoteAddress property, but not when I use the RemoteName property?

Frequently Asked Questions

The Asocket control establishes all connections by the RemoteAddress property. Since you don't always know the IP address but may know the host name of a computer, you can either use the GetHst control or the RemoteNameAddrXlate and the RemoteName property to translate the name to an IP.

When the RemoteNameAddrXlate property is used, you must set RemoteNameAddrXlate = True before setting the RemoteName property.

Note: When using RemoteNameAddrXlate, the control actually makes a UDP connection to resolve the IP, then attempts the user-specified connection. These two connections are done in series, which can be time consuming. Also, it is possible with some versions of winsock to set the RemoteAddress to a fully qualified domain name, but in general, you should always use an IP address in the RemoteAddress property and a host name in the RemoteName property.

What is the difference between a Datagram socket and a Stream socket?

Frequently Asked Questions

A Datagram socket is a UDP socket, while a Stream socket is a TCP socket. A socket simply consists of an IP address, the program's port number, and whether it uses TCP or UDP as the protocol.

When using TCP, a connection is established before data is transferred. TCP guarantees end-to-end integrity by insuring the packets are received by the remote program uncorrupted and are presented in the same order in which they were transmitted.

When using UDP, there is no connection established ahead of time --- the data is just sent out. Further, the UDP protocol does not guarantee the data will be received correctly, or received at all. This is often referred to as a "connectionless" connection.

The tradeoff between the two is speed and traffic. Obviously, TCP requires more overhead and time to establish the connection and handle the error checking.

Why won't my Internet Pack VBXes load into VB?

Frequently Asked Questions

The VBXes are looking for a file called WINSOCK.DLL. This DLL should be in your Windows directory (most DLLs are located in your Windows\System directory -- this one is an exception). Look for WINSOCK.DLL. If it's not in your Windows directory, we recommend moving it there. Be sure to write down where it was, in case something goes wrong.

Also, check the date on your WINSOCK.DLL. If it's 1994 or before, you should look into getting a later version.

Why do I get a GPF when I try to unload my form (or control) from the Done event?

Frequently Asked Questions

This is not uncommon in many controls. If the form containing the control is unloaded but the control's C++ code for the event has to reference the control, the GPF will occur because the control is no longer available after it is has been unloaded. The solution is to enable a timer in the Done event and have the Timer unload the form (or control).

With which TCP/IP stacks have your Internet controls been tested?

Frequently Asked Questions

The majority of our internal testing is done on either NT's or Win95's standard stacks. We also utilize a 3.1 machine running Trumpet Winsock.

As part of our beta program, the controls wind up on a variety of stacks like Novell (known to have differences in Winsock, but should be OK with the latest patches from Novell), WFWG (also has a known problem that can cause FTP trouble, but MS has a patch for that product as well (article ID Q122544)).

The controls support the standard Winsock interface, so in general, the 16-bit environments that do not come with a default stack (I.e., Windows 3.x) may involve a bit more setup, but as long as some reputable stack is used, there shouldn't be any problems.

How do I enable/disable the Windows 95 Dial-Up Networking connect prompt when my application issues a Connect method?

Frequently Asked Questions

The fact that the DUN pops up when attempting to establish a network connection is a Win95 OS setting. To change this behavior, choose Dial Up Networking from "My Computer", and select "Settings..." from the "Connections" menu. Set the desired value in the "When establishing a network connection" frame.

Why won't my Internet Pack VBX load?

Frequently Asked Questions

Usually, the Internet VBXes won't load when the WINSOCK.DLL is missing. Make sure you have a current WINSOCK.DLL in the Windows or Windows\System sub-dir. Some versions of Windows 3.x WinSocks may actually require a TCP/IP connection.

How do I convert my code from BLOCKING (Synchronous) to NON-BLOCKING (Asynchronous)?

Frequently Asked Questions

A quick fix for converting Blocking code to non-blocking code is as follows:

```
Blocking=False
```

In the Declarations of the Form, add:

```
Private fDone as Boolean
```

In the Done event of the control set the fDone flag as shown:

```
Private Sub FTP1_Done()
    fDone = True
End Sub
```

Then, when invoking a method, just loop until the Done event sets the fDone flag.

```
fDone = False
mMail1.Connect
Do
    DoEvents
    'here is where your application
    'can do other things
Loop Until (fDone)
```

Note: you may want to set a timer in the loop so it will not loop endlessly should some problem occur. Also, depending upon your code you may want to conditionally set the fDone flag in the AsyncError event.

How can I detect whether someone has entered an IP or host name?

Frequently Asked Questions

You can use the function (below) to check for a host name or IP address.

```
' This Function receives a string argument and
' validates whether the string is a valid IP value,
' by verifying that it is in the format of w.x.y.z and
' that each octet is between 0 and 255
'
' Returns True if IP there are 4 octets and each is
' between 0 and 255.
'
' Returns False in all other cases
'
' Disclaimer -- this function will not detect certain
' values such as netmasks like 255.255.255.255,
' which meet the criteria but are not valid IPs.
'
Private Function Valid_IP(IP As String) As Boolean
    Dim i As Integer
    Dim dot_count As Integer
    Dim test_octect As String

    IP = Trim$(IP)

    ' make sure the IP long enough before
    ' continuing
    If Len(IP) < 8 Then
        Valid_IP = False
        Exit Function
    End If

    i = 1
    dot_count = 0
    For i = 1 To Len(IP)
        If Mid$(IP, i, 1) = "." Then
            ' increment the dot count and
            ' clear the test octet variable
            dot_count = dot_count + 1
            test_octet = ""
            If i = Len(IP) Then
                ' we've ended with a dot
                ' this is not good
                Valid_IP = False
                Exit Function
            End If
        Else
            test_octet = test_octet & Mid$(IP, i, 1)
            On Error Resume Next
            byte_check = CByte(test_octet)
            If (Err) Then
                ' either the value is not numeric
                ' or exceeds the range of the byte
                ' data type.
                Valid_IP = False
                Exit Function
```

```
            End If
        End If
    Next i
    ' so far, so good
    ' did we get the correct number of dots?
    If dot_count <> 3 Then
        Valid_IP = False
        Exit Function
    End If
    ' we have a valid IP format!
    Valid_IP = True
End Function
```

What is the meaning of Error 20002 "unexpected server response"?

Frequently Asked Questions

The control has issued some command and the server did not accept it. It could be anything from an improperly formatted e-mail address to an unimplemented command on the server. You'll have to enable debugging to see what the command and reply are.

Can you recommend any good books that will help me understand Internet programming better?

Any good book on TCP/IP would be helpful. From personal experience, tech support recommends "TCP/IP" by Dr. Sidnie Feit, published by McGraw-Hill . It is *not* written from a programming standpoint, but does include everything you'd want to know about the lower levels of the OSI stack (including TCP/UDP/IP, etc.).

I'm unclear on blocking. Can you explain it to me?

When your application requests data from a network connection, it is hard to predict how long it will take before the data arrives and the call can complete. As a programmer, you have to determine whether to wait for the outcome of the call, or return immediately to your application and get the data *when* the data arrives.

Calls that wait, are called blocking calls. Because the call must complete before the application continues, blocking calls are also referred to as synchronous calls.

Calls that return control to your application immediately are called non-blocking calls. Since your application can perform tasks while the call is retrieving the data, non-blocking calls are also referred to as asynchronous calls.

Mabry Internet controls support both blocking and non-blocking calls.

It is important to note that even when using blocking calls, Windows can send event messages (such as Timer events, mouse clicks, etc.) to your application and it can respond to them. This can result in errors. It is the responsibility of the programmer to minimize the likelyhood of these situations (such as disabling any Timers or command buttons that will interrupt the call) and handle any errors should such conditions arise.

Error handling is very important when issuing calls to a network. Always use some method of On Error handling when invoking blocking calls. For non-blocking calls, normal On Error handling is required in addition to responding to the AsyncError event.

Should I use blocking or non-blocking calls?

Frequently Asked Questions

It depends on your application. See the explanation on blocking calls for a complete description of blocking vs. non-blocking.

Why do I keep getting the error "Busy executing asynchronous command"?

Frequently Asked Questions

A call has been invoked but a previous call has not been completed yet. Either set Blocking mode to true or wait for the Done event before issuing subsequent commands.

It is important to note that even when using blocking calls, Windows can send event messages (such as Timer events, mouse clicks, etc.) to your application and it can respond to them. This can result in errors. It is the responsibility of the programmer to minimize the likelyhood of these situations (such as disabling any Timers or command buttons that will interrupt the call) and handle any errors should such conditions arise.

Why do I keep getting errors when using an Internet VBX control?

A call has been invoked but a previous call has not been completed yet. Either set Blocking mode to true or wait for the Done event before issuing subsequent commands.

It is important to note that even when using blocking calls, Windows can send event messages (such as Timer events, mouse clicks, etc.) to your application and it can respond to them. This can result in errors. It is the responsibility of the programmer to minimize the likelihood of these situations (such as disabling any Timers or command buttons that will interrupt the call) and handle any errors should such conditions arise.

How do I tell what's happening when your control is talking to a server?

Frequently Asked Questions

The Internet Pack controls have debugging support built-in. Simply set the Debug property on the control to 1 and then add the following code to the Debug event of the control:

```
Debug.Print Message
```

Registration Information

CREDITS

ASocket 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.
Post Office Box 31926
Seattle, WA 98103-1926

Sales: 1-800-99-MABRY (U.S. Only)
Voice: 206-634-1443
Fax: 206-632-0272 or 206-364-3196
Web: http://www.mabry.com

COST

The price of ASocket (control only) is US$35 (US$40 for International orders). The cost of ASocket and the C/C++ source code (of the control itself) is US$90 (US$95 for International orders).

Prices are subject to change without notice.

DELIVERY METHODS

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

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

US$5.00	US Priority Mail
US$10.00	AirBorne Express 2nd Day (US deliveries only)
US$15.00	AirBorne Express Overnight (US deliveries only)
US$45.00	International AirBorne Express.

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). If you choose this option, please note: a printed manual is not included. We will, however, 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.

CompuServe E-Mail - CompuServe members can use the software registration forum (GO SWREG) to register this package. ASocket's SWREG ID number is 6389. The source code version's ID number is 9056. PLEASE NOTE: When you order through SWREG, we send the registered package to your CompuServe account (not your Internet or AOL account) within a few hours.

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), 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$12.50 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.

ASocket Order Form

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

Mail this form to:

Mabry Software, Inc.
Post Office Box 31926
Seattle, WA 98103-1926

Phone: 206-634-1443
Fax: 206-632-0272 or 206-364-3196
Internet: mabry@mabry.com
Web: www.mabry.com

Where did you get this copy of ASocket?

_____

Name: _____

Ship to: _____

_____

_____

_____

Phone: _____

Fax: _____

E-Mail: _____

MC/VISA/AMEX: _____ exp. _____

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

qty ordered ____ REGISTRATION
$35.00 ($40.00 international). Check or money order in U.S. currency drawn on a U.S. bank. Add $5.00 per order for shipping and handling.

qty ordered ____ SOURCE CODE AND REGISTRATION
$90.00 ($95.00 international). Check or money order in U.S. currency drawn on a U.S. bank. Add $5.00 per order for shipping and handling.

See Also

- **Bind** Method
- **Close** Method
- **Connect** Method
- **Create** Method
- **Listen** Method
- **Receive** Method
- **ReceiveFrom** Method
- **Send** Method
- **SendTo** Method

See Also

Connect Method
Listen Method
LocalAddress Property
LocalPort Property

See Also

Create Method

See Also

OnSend Event
Send Method

See Also

LingerEnabled Property
LingerMode Property
LingerTime Property

See Also

Action Property
Close Method
RemoteAddress Property
RemotePort Property

See Also

EventMask Property
LocalAddress Property
LocalPort Property
SocketType Property

See Also

Action Property

See Also

- **Create** Method
- **OnAccept** Event
- **OnClose** Event
- **OnConnect** Event
- **OnOutOfBandData** Event
- **OnReceive** Event
- **OnSend** Event

See Also

Connect Method

Create Method

See Also

Close Method
LingerMode Property
LingerTime Property

See Also

Close Method
LingerEnabled Property
LingerTime Property

See Also

Close Method
LingerEnabled Property
LingerMode Property

See Also

Create Method
LocalAddress Property
LocalPort Property

See Also

Listen Method

OnAccept Event

See Also

Connect Method

LocalPort Property

See Also

Connect Method

LocalAddress Property

See Also

Listen Method

See Also

Close Method

See Also

Connect Method

See Also

OutOfBandEnabled Property

ReceiveFrom Method

See Also

Receive Method

ReceiveFrom Method

See Also

Send Method
SendBuffer Property
SendTo Method

See Also

OnOutOfBandData Event

See Also

Action Property
OnReceive Event
ReceiveBuffer Property

See Also

Action Property
OnReceive Event
Receive Method
ReceiveFrom Method

See Also

ReceiveBuffer Property

SendBufferSize Property

See Also

RemoteAddress Property

RemotePort Property

See Also

Action Property
OnReceive Event
Receive Method
ReceiveBuffer Property
ReceiveFrom Method

See Also

RemoteName Property

RemoteNameAddrXlate Property

See Also

RemoteAddress Property

RemoteNameAddrXlate Property

See Also

RemoteAddress Property

RemoteName Property

See Also

LocalPort Property

RemoteAddress Property

See Also

LocalAddress Property

LocalPort Property

See Also

Connect Method

See Also

BytesSent Property
OnSend Event
SendBuffer Property

See Also

Action Property
OnSend Event
ReceiveBuffer Property
Send Method
SendTo Method

See Also

ReceiveBufferSize Property
SendBuffer Property

See Also

- **Action** Property
- **OnSend** Event
- **Send** Method
- **SendBuffer** Property
- **SendTo** Method

See Also

RemoteAddress Property
RemotePort Property
Send Method
SendBuffer Property

See Also

Listen Method

OnAccept Event

See Also

Bind Method
Connect Method
SocketPort Property

See Also

Bind Method
Connect Method
SocketAddress Property

See Also

Create Method

See Also

Connect Method

Action Property

See Also Frequently Asked Questions

Description

Setting the Action property invokes a specific socket action.

Syntax

*object.***Action** [= *integer*]

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

Part	Description
object	An ASocket control.
integer	An integer that specifies the action/method to perform.

Remarks

When using either the VBX or the ActiveX / OCX, you can assign one of the following values to the Action property to invoke the corresponding action. See the appropriate Methods for a description of each action. For instance, see Create for the effect of setting Action to ASocketCreate.

Constant	Value	Description
ASocketCreate	1	Create method
ASocketBind	2	Bind method
ASocketClose	3	Close method
ASocketConnect	4	Connect method
ASocketListen	5	Listen method
ASocketReceive	6	Receive method
ASocketReceiveFrom	7	ReceiveFrom method
ASocketSend	8	Send method
ASocketSendTo	9	SendTo method

Data Type

Integer

Bind Method

See Also Frequently Asked Questions

Description

Associates a local address with the socket.

Syntax

*object.***Bind** *LocalAddress*, *LocalPort*

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

Part	Description
object	Required. An ASocket control.
LocalAddress	Optional. A string expression that specifies the address of the server to listen to. If not specified, the LocalAddress property is used.
LocalPort	Optional. An integer that specifies the port number, on the server, to listen to. If not specified, the LocalPort property is used.

Remarks

This method is used on an unconnected datagram or stream socket, before subsequent Connect or Listen calls. Before it can accept connection requests, a listening server socket must select a port number and make it known to Windows Sockets by calling Bind. Bind establishes the local association (host address/port number) of the socket by assigning a local name to an unnamed socket.

BroadcastEnabled Property

See Also Frequently Asked Questions

Description

Enables the transmission of broadcast packets.

Syntax

*object.***BroadcastEnabled** [= *boolean*]

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

Part	Description
object	An ASocket control.
boolean	A boolean expression that indicates whether packets may be broadcast.

Remarks

Broadcast packets can only be sent over datagram sockets.

Data Type

Integer (boolean)

BytesSent Property

See Also Frequently Asked Questions

Description

Holds the number of bytes sent with the Send method.

Syntax

*object.***BytesSent**

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

Part	Description
object	An ASocket control.

Remarks

After the Send method is called, the BytesSent property will contain the number of bytes queued for transmission by the socket protocol stack. If the protocol stack queue is either full when the Send method is called, or becomes full as a result of some of the Send data being queued, then BytesSent can be examined to determine how many bytes remain in the SendBuffer to be sent.

Microsoft's WinSock stack does not queue partial messages and so, if the Send method fails, the WSAEWOULDBLOCK error (10035) is thrown and BytesSent will be zero. However, in the future Microsoft may decide to queue some of the bytes in the SendBuffer property. And, of course, protocol stacks from other vendors may do so as well. If only some of the bytes are sent, then an error is thrown with the value of the Err variable set to 20000. BytesSent will be some value greater than zero and less than the number of bytes in SendBuffer. In this case, you will need to wait for the OnSend event and then send the remainder of the bytes. Be sure to do something like:

```
Ctl.SendBuffer = right(len(Ctl.SendBuffer) - Ctl.BytesSent)
Ctl.Send
```

Data Type

Integer (long)

Close Method

See Also Frequently Asked Questions

Description

Closes a created socket.

Syntax

*object.***Close**

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

Part	Description
object	Required. An ASocket control.

Remarks

The semantics of Close are affected by the LingerEnabled, LingerMode, and LingerTime properties.

If the LingerEnabled property is False, then Close will return immediately. However, any data queued for transmission will be sent, if possible, before the underlying socket is closed. This is also called a graceful disconnect.

If LingerEnabled is True and if LingerMode is non-zero, the socket will close gracefully. Control will return from the call after the socket finishes closing, if LingerTime is non-zero.

If LingerMode is zero or LingerTime is zero, the socket will be closed immediately.

Connect Method

See Also Frequently Asked Questions

Description

Establishes a connection to a peer.

Syntax

*object.***Connect** *RemoteAddress*, *RemotePort*

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

Part	Description
object	Required. An ASocket control.
RemoteAddress	Optional. A string expression that specifies the address of the server to connect to. If not specified, the RemoteAddress property is used.
RemotePort	Optional. An integer that specifies the port number, on the server, to connect to. If not specified, the RemotePort property is used.

Remarks

The Connect method is used to establish a connection to a server. The *RemotePort* and *RemoteAddress* parameters (or properties, when the optional parameters are not passed) specify the server to connect to.

Connect is an asynchronous operation and the Connect method will most likely return a WSAEWOULDBLOCK (10035) error. When connecting to local server ports, this error may not occur. If the WSAEWOULDBLOCK error is returned, you should wait until the OnConnect event is fired before attempting to communicate with the server socket.

Note: VBX users must set the RemotePort and RemoteAddress properties prior to using the Action property to invoke the Connect method. ActiveX / OCX users may either set the properties or pass them as arguments. Passed arguments override the contents of the RemotePort and RemoteAddress properties.

Create Method

See Also Frequently Asked Questions

Description

Creates and opens a socket.

Syntax

*object.***Create** *LocalPort*, *SocketType*, *EventMask*, *LocalAddress*

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

Part	Description
object	Required. An ASocket control.
LocalPort	Optional. An integer that specifies the port number to create.
SocketType	Optional. An integer that specifies the type of socket: stream or datagram.
EventMask	Optional. An integer that specifies the events to fire for this socket.
LocalAddress	Optional. A string expression that specifies the address of the socket to create.

Remarks

The Create method creates and opens a socket using the specified port and IP address.

SocketType can be:

Constant	Value	Description
ASocketStream	0	Stream socket
ASocketDatagram	1	Datagram socket

EventMask can be any combination of the following:

Constant	Value	Description
ASocketReadEvent	1	OnReceive event fired when data is received.
ASocketWriteEvent	2	OnSend event fired when the socket is ready to send data.
ASocketOOBEvent	4	OnOutOfBandData event fired when out-of-band data is received.
ASocketAcceptEvent	8	OnAccept event fired when a client connection has been accepted.
ASocketConnectEvent	16	OnConnect event fired when a socket connects to a server.
ASocketCloseEvent	32	OnClose event fired when the socket closes.
ASocketAllEvents	63	Enables all of the socket events.

ErrorNumber Property

See Also Frequently Asked Questions

Description

Returns last error code.

Syntax

*object.***ErrorNumber**

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

Part	Description
object	An ASocket control.

Remarks

This property holds the most recent error number returned by this control. Any time an Action is performed or any method is executed, this property is reset to zero (0) prior to executing the action or method.

Data Type

Integer

EventMask Property

See Also Frequently Asked Questions

Description

Enables the socket control's notification events.

Syntax

*object.***EventMask** [= *integer*]

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

Part	Description
object	An ASocket control.
integer	An integer that specifies the events fired for the current socket.

Remarks

One or more of the following constants can be used to enable socket events:

Constant	Value	Description
ASocketReadEvent	1	OnReceive event fired when data is received.
ASocketWriteEvent	2	OnSend event fired when the socket is ready to send data.
ASocketOOBEvent	4	OnOutOfBandData event fired when out-of-band data is received.
ASocketAcceptEvent	8	OnAccept event fired when a client connection has been accepted.
ASocketConnectEvent	16	OnConnect event fired when a socket connects to a server.
ASocketCloseEvent	32	OnClose event fired when the socket closes.
ASocketAllEvents	63	Enables all of the socket events.

Data Type

Integer

KeepAliveEnabled Property

See Also Frequently Asked Questions

Description

Enables the transmission of keep-alive packets.

Syntax

*object.***KeepAliveEnabled** [= *boolean*]

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

Part	Description
object	An ASocket control.
boolean	A boolean expression that indicates whether keep-alive packets are sent to the server.

Remarks

Some servers will drop a connection for inactivity. The KeepAliveEnabled property can be set to True to cause keep-alive packets to be sent to the server.

Data Type

Integer (boolean)

LingerEnabled Property

See Also Frequently Asked Questions

Description

Affects the way a socket is closed.

Syntax

*object.***LingerEnabled** [= *boolean*]

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

Part	Description
object	An ASocket control.
boolean	A boolean expression that indicates whether the socket lingers prior to closing.

Remarks

Please see the Close method for a description of how this property affects socket closing.

Data Type

Integer (boolean)

LingerMode Property

See Also Frequently Asked Questions

Description

LingerMode affects the way the Close method works.

Syntax

*object.***LingerMode** [= *integer*]

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

Part	Description
object	An ASocket control.
integer	An integer that determines if lingering is on or off.

Remarks

Please see the Close method for details on the use of this property. This property may be set to the following values:

Constant	Value	Description
AsocketLingerModeOff	0	Linger mode is disabled.
ASocketLingerModeOn	1	Linger mode is enabled.

Data Type

Integer

LingerTime Property

See Also Frequently Asked Questions

Description

The LingerTime property affects the way the Close method works.

Syntax

*object.***LingerTime** [= *time*]

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

Part	Description
object	An ASocket control.
time	An integer that determines how long the socket lingers while closing.

Remarks

Please see the Close method for details on the use of this property.

Data Type

Integer

Listen Method

See Also Frequently Asked Questions

Description

This method enables a socket to listen for incoming connection requests.

Syntax

*object.***Listen**

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

Part	Description
object	Required. An ASocket control.

Remarks

Listen is the method to use if you want to accept client connection requests. In other words, this is how you create a server. Listen applies only to sockets that support connections, that is, those of type ASocketStream (see the Create method). This socket is put into passive mode where incoming connections are acknowledged and queued, pending acceptance by the process.

Listening Property

See Also Frequently Asked Questions

Description

Returns True when a socket is waiting (listening) for a client to connect.

Syntax

*object.***Listening**

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

Part	Description
object	An ASocket control.

Remarks

When a socket is waiting for a client to connect, it is listening, and the Listening property will be true.

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

Data Type

Integer (boolean)

LocalAddress Property

See Also Frequently Asked Questions

Description

Specifies the IP address of the socket.

Syntax

*object.***LocalAddress** [= *string*]

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

Part	Description
object	An ASocket control.
string	A string expression that specifies an IP address.

Remarks

Before connecting to a remote server you must set the LocalAddress and LocalPort properties of the socket to appropriate values.

Data Type

String

LocalPort Property

See Also Frequently Asked Questions

Description

Specifies the socket's port number.

Syntax

*object.***LocalPort** [= *integer*]

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

Part	Description
object	An ASocket control.
integer	An integer that specifies a port.

Remarks

Before connecting to a remote server you must set the LocalAddress and LocalPort properties of the socket to appropriate values.

Data Type

Integer

OnAccept Event

See Also Frequently Asked Questions

Description

The OnAccept event is fired when a client connects to a listening socket.

Syntax

Sub *object*_**OnAccept(**[*index* **As Integer**,] *SocketHandle* **As Integer**, *ErrorCode* **As Long)**

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

Part	Description
object	An ASocket control.
index	An integer that identifies a control if it's in a control array.
SocketHandle	An integer that specifies the handle of the new socket.
ErrorCode	A long integer that specifies an error, if any.

Remarks

To establish a connection with a client, you need to have at least two sockets, one to Listen for client connections and another which is used to establish the client connection. If you have two socket controls named Server and ClientConnection, then the following code will establish a connection with a client:

```
Sub Server_OnAccept (SocketHandle As Long, ErrorCode As Integer)
    ClientConnection.Socket = SocketHandle
End Sub
```

If you want to support multiple clients simultaneously, you will need to use either a control array, or multiple forms, each with a socket control. See the SERVER3 sample project for an example of a multiple-client server.

OnClose Event

See Also Frequently Asked Questions

Description

The OnClose event is fired when the socket is closed.

Syntax

Sub *object*_**OnClose(**[*index* **As Integer**,] *ErrorCode* **As Long)**

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

Part	Description
object	An ASocket control.
index	An integer that identifies a control if it's in a control array.
ErrorCode	A long integer that specifies an error, if any.

Remarks

This event fires when the socket closes. *ErrorCode* gives the most recent error on the closed socket.

OnConnect Event

See Also Frequently Asked Questions

Description

Fires when a connection is complete.

Syntax

Sub *object*_**OnConnect(**[*index* **As Integer**,] *ErrorCode* **As Long)**

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

Part	Description
object	An ASocket control.
index	An integer that identifies a control if it's in a control array.
ErrorCode	A long integer that specifies an error, if any.

Remarks

The OnConnect event is fired when the Connect method finishes connecting to a server socket.

OnOutOfBandData Event

See Also Frequently Asked Questions

Description

Fired when out-of-band data is received.

Syntax

Sub *object*_**OnOutOfBandData(**[*index* **As Integer**,] *ErrorCode* **As Long)**

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

Part	Description
object	An ASocket control.
index	An integer that identifies a control if it's in a control array.
ErrorCode	A long integer that specifies an error, if any.

Remarks

Out-of-band data is a logically independent channel that is associated with each pair of connected stream sockets. Datagram sockets do not support out-of-band-data. The channel is generally used to send urgent data. To retrieve the data, use the ReceiveFrom method.

OnReceive Event

See Also Frequently Asked Questions

Description

Fired when the socket has data available to be read.

Syntax

Sub *object*_**OnReceive(**[*index* **As Integer**,] *ErrorCode* **As Long)**

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

Part	Description
object	An ASocket control.
index	An integer that identifies a control if it's in a control array.
ErrorCode	A long integer that specifies an error, if any.

Remarks

When data arrives at a socket the OnReceive event is fired.

The data can be read using the Receive method as in the following code:

```
Sub AsyncSocket1_OnReceive (ErrorCode As Integer)
    Dim t As String
    '
    ' Receive data available, get it
    '
    AsyncSocket1.Action = ASocketReceive
    t = AsyncSocket1.ReceiveBuffer
    '
    ' Echo to client
    '
    AsyncSocket1.SendBuffer = t
    AsyncSocket1.Action = ASocketSend
End Sub
```

OnSend Event

See Also Frequently Asked Questions

Description

The OnSend event is fired when the socket connection is ready to send data.

Syntax

Sub *object*_**OnSend(**[*index* **As Integer**,] *ErrorCode* **As Long)**

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

Part	Description
object	An ASocket control.
index	An integer that identifies a control if it's in a control array.
ErrorCode	A long integer that specifies an error, if any.

Remarks

The OnSend event is fired once after the socket is connected. It is not fired each time you send data. After the OnSend event is received, you can send as many packets as you like, or until an error occurs, without waiting for another OnSend event.

Note that if the output queue becomes full (this may happen if you're sending lots of data), an error will be returned from the Send method. Then, when the socket is ready to send data again, the OnSend event will be fired.

One way to think of the OnSend event is that it is fired whenever the socket is ready for you to send data after some period of time when it was not ready. When a socket is first connected, it is not ready for you to send data. When it becomes ready, the OnSend event is fired. Then, if later sends overflow the output queue, the OnSend event will be fired after the socket once again becomes ready for data.

OutOfBandEnabled Property

See Also Frequently Asked Questions

Description

Enables the receipt of out-of-band data on a stream socket.

Syntax

*object.***OutOfBandEnabled** [= *boolean*]

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

Part	Description
object	An ASocket control.
boolean	A boolean expression that indicates whether out of band data is enabled.

Remarks

Each pair of connected stream sockets (a client and a server, for instance) has a logically independent channel over which out-of-band data can be transmitted. This channel provides for unbuffered transmission of one packet of data which is typically of an urgent nature. Setting the OutOfBandEnabled flag to True enables the socket to receive these messages.

Data Type

Integer (boolean)

ReceiveBufferSize Property

See Also Frequently Asked Questions

Description

Sets size of the socket's receive buffer.

Syntax

*object.***ReceiveBufferSize** [= *integer*]

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

Part	Description
object	An ASocket control.
integer	An integer that specifies the receive buffer size.

Remarks

Used to change the default receive buffer size. The default receive buffer size should be adequate for most purposes.

Note: this property specifies the size of the buffer used internally by WinSock and has nothing to do with the ReceiveBuffer property which will simply hold as much data as was received by any read.

Data Type

Integer

Receive Method

See Also Frequently Asked Questions

Description

Used to retrieve receive data from the socket control.

Syntax

*object.***Receive**

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

Part	Description
object	Required. An ASocket control.

Remarks

When working with the VBX, you will need to use the Action property and ReceiveBuffer property. If you're working with the ActiveX / OCX and want to be compatible with the VBX (for porting reasons), you can also use the Action and ReceiveBuffer properties. For example:

```
AsyncSocket1.Action = ASocketReceive
str = AsyncSocket1.ReceiveBuffer
```

For the ActiveX / OCX version you can also use the Receive method directly as in:

```
str = AsyncSocket1.Receive()
```

ReceiveBuffer Property

See Also Frequently Asked Questions

Description

Contains received data after the Receive method is invoked.

Syntax

*object.***ReceiveBuffer** [= *string*]

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

Part	Description
object	An ASocket control.
string	A string expression that contains the data received.

Remarks

Data received from a remote connected socket is accessed through the ReceiveBuffer property.

Data Type

String

ReceiveFrom Method

See Also Frequently Asked Questions

Description

Receives a datagram and stores the source address.

Syntax

*object.***ReceiveFrom** *RemoteAddress*

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

Part	Description
object	Required. An ASocket control.
RemoteAddress	Optional. A string expression that specifies the address of the socket that is sending data.

Remarks

When reading from a datagram socket, or when reading out-of-band-data, you should use the ReceiveForm method. The data is stored in the ReceiveBuffer and the sending sockets address and port number are returned in the RemoteAddress and RemotePort and in the corresponding control properties.

ReceiveTimeout Property

See Also Frequently Asked Questions

Description

Specifies the number of seconds to wait to receive data.

Syntax

*object.***ReceiveTimeout** [= *integer*]

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

Part	Description
object	An ASocket control.
integer	An integer that specifies a duration, in seconds.

Remarks

If you set this property to a non-zero value then, after that number of seconds has elapsed while waiting for data, an error will be returned.

Data Type

Integer

RemoteAddress Property

See Also Frequently Asked Questions

Description

Address of remote socket (i.e., 123.123.12.3).

Syntax

*object.***RemoteAddress** [= *string*]

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

Part	Description
object	An ASocket control.
string	A string expression that specifies an IP address.

Remarks

Contains the IP address of the remote socket to connect to.

Data Type

String

RemoteName Property

See Also Frequently Asked Questions

Description

Name of remote socket (i.e., ftp.microsoft.com).

Syntax

*object.***RemoteName** [= *string*]

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

Part	Description
object	An ASocket control.
string	A string expression that specifies a host name.

Remarks

When the RemoteNameAddrXlate property is set to True, setting the RemoteName property will attempt to determine the corresponding IP address and set the RemoteAddress property accordingly.

Note that, with WinSock, it is not possible to do this both quickly and reliably. The ASocket control performs the lookup in such a way that it will not hang for 2-3 minutes if the computer is not connected to the net. However, the trade-off for this is that it may happen occasionally that the lookup fails, even though the network is actually accessible. For completely reliable name lookups, you should use the GetHst control.

Data Type

String

RemotePort Property

See Also Frequently Asked Questions

Description

Specifies the port address of the remote socket.

Syntax

*object.***RemotePort** [= *integer*]

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

Part	Description
object	An ASocket control.
integer	An integer that specifies a port.

Remarks

Together RemotePort and RemoteAddress fully specify a remote socket's address.

Data Type

Integer

RemoteNameAddrXlate Property

See Also Frequently Asked Questions

Description

Used to translate names, such as "mabry.com", to IP addresses (204.157.98.11).

Syntax

*object.***RemoteNameAddrXlate** [= *boolean*]

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

Part	Description
object	An ASocket control.
boolean	A boolean expression that indicates whether host names will be translated into IP addresses.

Remarks

When RemoteNameAddrXlate is True, changes to either RemoteName or RemoteAddress will initiate an attempt to use a DNS (domain name server) to translate one to the other. For instance, if RemoteName is set to "activexpert.com", RemoteNameAddrXlate is True, and if the local machine is connected to the TCP/IP network, the control will attempt to translate "activexpert.com" to its IP address "206.161.236.182".

If the attempted translation fails (between either RemoteAddress or RemoteName) because the local machine is not attached to the TCP/IP network, or it cannot reach its DNS, the other property is left unchanged. If an error occurs for any other reason, the other property is set to an empty string.

The following sequence can be used to obtain a host's address:

```
AsyncSocket1.RemoteNameAddrXlate = FALSE
AsyncSocket1.RemoteAddress = "0.0.0.0"
AsyncSocket1.RemoteNameAddrXlate = TRUE
AsyncSocket1.RemoteName = "activexpert.com"
If (AsyncSocket1.RemoteAddress <> "" ) Then
    Debug.Print AsyncSocket1.RemoteAddress
EndIf
```

Please see the Remarks section of the RemoteName property for further comments on the use of this property.

Data Type

Integer (boolean)

ReuseAddressEnabled Property

See Also Frequently Asked Questions

Description

Enables other sockets to use the same address as this socket.

Syntax

*object.***ReuseAddressEnabled** [= *boolean*]

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

Part	Description
object	An ASocket control.
boolean	A boolean expression that indicates whether socket addresses may be re-used.

Remarks

By default, a socket cannot use a local address which is already in use. On occasion, however, it may be desirable to reuse an address in this way. Since every connection is uniquely identified by the combination of local and remote addresses, there is no problem with having two sockets use the same local address as long as the remote addresses are different.

The ReuseAddressEnabled property must be set to True before invoking an Action command. Otherwise, it has no effect.

Data Type

Integer (boolean)

RouteEnabled Property

See Also Frequently Asked Questions

Description

Enables packet routing.

Syntax

*object.***RouteEnabled** [= *boolean*]

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

Part	Description
object	An ASocket control.
boolean	A boolean expression that indicates whether packet routing is enabled.

Remarks

When this property is set to True, packet routing is enabled.

Data Type

Integer (boolean)

Send Method

See Also Frequently Asked Questions

Description

Sends a buffer of data to the connected socket.

Syntax

*object.***Send** *SendBuffer*

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

Part	Description
object	Required. An ASocket control.
SendBuffer	Optional. A string expression that specifies the data to send. If not specified, the SendBuffer property is sent.

Remarks

Sends the contents of the SendBuffer property to the connected socket. When using the ActiveX / OCX, you can supply *SendBuffer* as an argument and it will be sent instead of the contents of the SendBuffer property.

There are two special situations to watch out for when sending data. Both are signaled by firing an error during execution of the Send method. For this reason, you will need to be sure to have error handling code in place where your code sends data over a socket.

No matter what protocol stack is in use on a machine, there will be a limit as to how much data may be queued for transmission at any one time. When the output queue is full (or nearly full) and you call the Send method, either a 10035 (WSAEWOULDBLOCK) error will be fired, or a 20000 (could not send all data) error will be fired. The first error (10035) tells you that none of the data was queued. The second error (20000) tells you that some of the data was queued. The BytesSent property can then be examined to determine how much data was sent. In either case, you must wait until the next OnSend event occurs before continuing to send data over the socket.

SendBuffer Property

See Also Frequently Asked Questions

Description

Buffer for data to be sent to the remote socket.

Syntax

*object.***SendBuffer** [= *string*]

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

Part	Description
object	An ASocket control.
string	A string expression that contains the data to send.

Remarks

Data to send to connected socket.

Data Type

String

SendBufferSize Property

See Also Frequently Asked Questions

Description

Sets the size of the socket's send buffer.

Syntax

*object.***SendBufferSize** [= *integer*]

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

Part	Description
object	An ASocket control.
integer	An integer that specifies the send buffer size.

Remarks

Used to change the default transmit buffer size.

Note: this property specifies the size of the buffer used internally by Winsock and has nothing to do with the SendBuffer property.

Data Type

Integer

SendTimeout Property

See Also Frequently Asked Questions

Description

Specifies the number of seconds to wait to send data.

Syntax

*object.***SendTimeout** [= *integer*]

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

Part	Description
object	An ASocket control.
integer	An integer that specifies a duration, in seconds.

Remarks

If you set this property to a non-zero value, after that number of seconds has elapsed while waiting to send data, an error will be returned.

Data Type

Integer

SendTo Method

See Also Frequently Asked Questions

Description

Sends data to a specific remote socket.

Syntax

*object.***SendTo** *RemoteAddress*, *SendBuffer*

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

Part	Description
object	Required. An ASocket control.
RemoteAddress	Optional. A string expression that specifies the address of the socket to send to.
SendBuffer	Optional. A string expression that specifies the data to send. If not specified, the SendBuffer property is sent.

Remarks

SendTo is used to send the contents of *SendBuffer* (using a socket of type ASocketDatagram) to another socket as specified by the RemoteAddress and RemotePort properties.

Socket Property

See Also Frequently Asked Questions

Description

Windows socket handle.

Syntax

*object.***Socket** [= *long*]

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

Part	Description
object	An ASocket control.
long	A long integer that specifies a socket handle.

Remarks

This property exists solely for the purpose of accepting a client connection request. The OnAccept event passes a Socket handle as one of its parameters. That parameter must be assigned to the Socket property of some control other than the one firing the OnAccept event, in order to accept the connection.

Data Type

Long

SocketAddress Property

See Also Frequently Asked Questions

Description

The local socket's IP address.

Syntax

*object.***SocketAddress** [= *string*]

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

Part	Description
object	An ASocket control.
string	A string expression that specifies an IP address.

Remarks

This property is especially useful when connecting to the Internet through an ISP which dynamically assigns IP addresses. After a remote socket connection has been established, this property will return the IP address assigned to the socket. This property is also useful when a Connect call has been made without doing a Bind first; this provides the only means by which you can determine the local association which has been set by the system.

Data Type

String

SocketPort Property

See Also Frequently Asked Questions

Description

Local port number.

Syntax

*object.***SocketPort** [= *integer*]

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

Part	Description
object	An ASocket control.
integer	An integer that specifies a port.

Remarks

Port number used by the socket. This property is valid only after a remote system is connected.

Data Type

Integer

SocketType Property

See Also Frequently Asked Questions

Description

Determines whether a socket will be a stream or datagram socket.

Syntax

*object.***SocketType** [= *integer*]

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

Part	Description
object	An ASocket control.
integer	An integer that determines the style of socket: stream or datagram.

Remarks

The SocketType property may be one of the following:

Constant	Value	Description
AsocketStream	0	Provides sequenced, reliable, full-duplex, connection-based byte streams.
ASocketDatagram	1	Supports datagrams, which are connectionless, unreliable packets of a fixed (typically small) maximum length.

Data Type

Integer (enumerated)

TCPNoDelayEnabled Property

See Also Frequently Asked Questions

Description

Disables the Nagle delay algorithm when set to True.

Syntax

*object.***TCPNoDelayEnabled** [= *boolean*]

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

Part	Description
object	An ASocket control.
boolean	A boolean expression that indicates whether the Nagle delay algorithm is disabled.

Remarks

The NoDelayEnabled property enables the Nagle algorithm. The Nagle algorithm is used to reduce the number of small packets sent by a host by buffering unacknowledged send data until a full-size packet can be sent. However, for some applications this algorithm can impede performance, and NoDelayEnabled be used to turn it off. You should not set this property to True unless the impact of doing so is well-understood and desired.

Data Type

Integer (boolean)

Version Property

Frequently Asked Questions

Description

Shows the version of the control.

Syntax

*object.***Version**

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

Part	Description
object	An ASocket 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.
Post Office Box 31926
Seattle, WA 98103-1926

Phone: 206-634-1443
Fax: 206-632-0272 or 206-364-3196

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.