SocketWrench™ Custom Control
Technical Reference

Version 1.0

Copyright © 1995, Catalyst Software. All rights reserved.
SocketWrench is a trademark of Catalyst Software.
Microsoft is a registered trademark, and Windows and Visual Basic are trademarks of Microsoft Corporation.

Introduction

The SocketWrench custom control provides network communication services for your Visual Basic application using a third-party TCP/IP protocol stack and compatible Windows Sockets library. The control offers the following features:

- Client and server functionality in a single control
- Support for blocking and non-blocking socket operations
- Convenient "text" mode for receiving data a line at a time
- Support for both stream and datagram protocols
- Send and receive urgent (out-of-band) data

Instead of using API calls, virtually all socket functions can be performed by setting control properties and responding to events. For those developers who are not familiar with the details of socket programming, SocketWrench can also insulate them from many of the common pitfalls, without sacrificing functionality or flexibility.

Each control that you use corresponds to one socket (which may or may not be connected to a remote host). If you need access to multiple sockets, you must use multiple controls, typically as a control array. This is most commonly needed when your application acts a server and must be able to handle several connections at one time.

Blocking vs. Non-Blocking Sockets

When designing your application, it is important to make some decisions about how SocketWrench will be used. There are three general approaches that can be taken with the control:

- Use a blocking (synchronous) socket. In this mode, the program will not resume execution until the socket operation has completed. This method is only recommended for relatively small applications. Because blocking sockets must allow messages to be processed in Windows, it is possible that your blocked application can be re-entered at a different point. This approach can lead to complex interactions (and difficult debugging) if there are multiple active controls in use by the application.

- Use a non-blocking (asynchronous) socket, which allows your application to respond to events. For example, when the remote system writes data to the socket, a Read event is generated for the control. Your application can respond by reading the data from the socket, and perhaps send some data back, depending on the context of the data received.

- Use a combination of blocking and non-blocking socket operations. The ability to switch between blocking and non-blocking modes "on the fly" provides a powerful and convenient way to perform socket operations. Note that the warning regarding blocking sockets also applies here.

If you decide to implement your application using a blocking socket connection, it may call a *blocking hook function* that will allow your application to be re-entered. The Windows Sockets API will fail most socket operations while a blocking socket call is in progress, so your application must be prepared to handle this. The **State** property can be used to determine the state of a blocking socket.

Buffering Socket Data

SocketWrench may be configured to buffer data transfers between your application and the remote host to minimize the number of calls made to the Windows Sockets library. Buffering can only be enabled for stream sockets by setting the **BufferSize** property before the socket is created. Both blocking and non-blocking sockets can be buffered.

Binary vs. Text Mode

Buffered sockets can receive data in either binary or text mode, depending on the value of the **Binary** property. By default, all data read from the socket is returned to your application exactly as it was sent by the remote host. In some cases, however, it would be more convenient to receive data as lines of text, similar to reading a text file. By setting the **Binary** property to **False**, each read on the socket is terminated when a linefeed is encountered in the data stream (if a carriage-return precedes the linefeed, it is discarded). Keep in mind that this only affects how data is received by your application and has no effect on data being written to the socket.

Server Implementation

In addition to using SocketWrench to connect to remote hosts as a client, you can use the same control to create a server application. To do this, you create a *listening* socket which waits for connections from clients. When a client connects to your socket, you are notified via the **Accept** event (this event is only generated for non-blocking sockets). At this point, you may either accept the connection or close the listening socket. When accepting a connection, you have two options:

- Use the SOCKET_ACCEPT action on the listening socket to accept the connection. Since this action closes the listening socket, any other clients attempting to connect to your server will be rejected.

- Set the **Action** property of an unused socket to the handle of the listening socket. This will allow the unused socket to accept the connection on behalf of the listening socket, which will continue to listen for additional connections.

Of the two approaches, the first is simpler and suitable for small, special purpose servers that do not need to handle multiple client connections. The second approach is more powerful, but also more complex. To handle a variable number of client connections, the recommended approach is to create a control array of sockets that can be loaded and unloaded as needed.

Commonly Used Properties

Although SocketWrench has a large number of properties, only a subset of them will be used with any frequency in your applications. Here are the properties that you should become familiar with first:

Property	Description
Action	Determines the action to be taken on the socket. The various actions that can be taken on the socket are defined in the CSWSOCK.BAS file as global constants.
Binary	Determines how data is read from the socket. If set to False, then data is returned as text terminated by newlines. The socket must be buffered and connected using a streams-based protocol (i.e.: TCP)

Property	Description
Blocking	Set or return the blocking mode for the socket. By default, sockets are blocking, and no socket events will be generated by the control.
HostAddress	Specifies the IP address of the remote host that the socket will be sending data to or receiving data from. Setting this property will automatically update the HostName property.
HostName	Specifies the name of the remote host. Setting this property will automatically update the HostAddress property.
Protocol	Specifies the protocol to use for this socket. A value of 0 indicates that the appropriate protocol for the socket type selected is used.
RecvData	A read-only property that returns data to be read from the socket. This property uses the RecvLen property to determine the maximum number of bytes to read.
RecvLen	Specifies the number of bytes to read from the socket. After reading the RecvData property, this property is changed to reflect the number of bytes actually read.
RemotePort	Specifies the port number that a remote server is listening. Setting this value causes the RemoteServer property to be updated.
RemoteService	Specifies the service name of the remote port a server is listening to. Setting this value causes the RemotePort property to be updated.
SendData	A write-only property that is used to write data to the socket. The SendLen property specifies the maximum number of bytes that may be written.
SendLen	Specifies the maximum number of bytes that may be written to the socket. After sending the data, this property is changed to reflect the number of bytes actually written.
Type	Specifies the type of socket to be created. The type will commonly be either SOCK_STREAM or SOCK_DGRAM. These values are defined as global constants in the CSWSOCK.BAS file.

Requirements

The sockets control requires Microsoft Windows 3.1 or later, Visual Basic 2.0 or later and a TCP/IP product that supports the Windows Sockets 1.1 specification. The WINSOCK.DLL library must be located in the Windows directory or in a directory that is in the search path.

When you distribute your application that uses SocketWrench, you should install the CSWSOCK.VBX file in the Windows system directory. Note that it is not required that the end-user of your application use the same TCP/IP product that you've used for development.

Accept Property

Description A write-only property, that when set to the value of a listening socket's handle, accepts the connection. The listening socket is not closed and may continue to listen for new client connections.

Visual Basic [*form*.]*Socket*.**Accept** = *handle*%

Remarks Setting this property to a socket that is not listening for connections will generate an error. If the Accept property is set to the value of the current socket's handle, the effect is the same as if the SOCKET_ACCEPT action has been taken.

Data Type **Integer**

See Also Action Property, Backlog Property, Listening Property, Accept Event

Action Property

Description Setting this property causes the socket to take some action, such as creating a socket, connecting to a remote system or closing a connection.

Visual Basic [*form*.]*Socket*.**Action** = *action*%

Remarks The following table lists the actions that a socket control may take:

Value	Constant	Description
1	SOCKET_OPEN	Create a socket using the AddressFamily, Protocol and Type properties. This action is commonly used to create datagram sockets, or if supported by the TCP vendor, raw sockets.
2	SOCKET_CONNECT	Connect to a remote system specified by the HostAddress or HostName properties. If a socket has not already been created, this action will create it.
3	SOCKET_LISTEN	Listen on a socket for incoming connections on the port specified by the LocalPort or LocalService properties. If a socket has not already been created, this action will create it.
4	SOCKET_ACCEPT	Accept an incomming connection on the socket. The listening socket connection is closed and the client is connected. To resume listening for new connections, the Action must be set to SOCKET_LISTEN again. This approach allows only one incomming connection per listening socket.
5	SOCKET_CANCEL	Cancel the current blocking operation. This action only has meaning for blocking sockets.
6	SOCKET_FLUSH	Flush the contents of the send and receive socket buffers.
7	SOCKET_CLOSE	Close the socket, and if connected, break the connection with the remote system. This action should be taken before the control is unloaded from the form.
8	SOCKET_ABORT	Immediately close the socket without waiting for remaining data to be written out. This action should only be taken under extreme circumstances.

Action Property

The SOCKET_CONNECT and SOCKET_ACCEPT actions are always blocking (i.e.: the value of the Blocking property is ignored) and will use the Timeout property to determine the amount of time to wait until the action times-out.

Data Type **Integer**

See Also Connected Property, Listening Property, State Property

AddressFamily Property

Description Sets or returns the address family for the socket.

Visual Basic [*form*.]*Socket*.**AddressFamily** [= *aftype*%]

Remarks The address family value should be set to the constant AF_INET. This property is included only for completeness and future expansion.

Data Type **Integer**

See Also Protocol Property, Type Property

AtMark Property

Description A read-only property that returns True if the next receive will return urgent data.

Visual Basic [*form*.]*Socket*.**AtMark**

Remarks This property can only be used if the socket type is SOCK_STREAM and the InLine property has been set to True. Reading this property is the same as using the SIOCATMARK option with the **ioctlsocket** function.

Data Type **Integer** (Boolean)

See Also RecvData Property, Urgent Property, Read Event

Backlog Property

Description Sets or returns the number of connections that may be accepted by a listening socket.

Visual Basic [*form*.]*Socket*.**Backlog** [= *backlog*%]

Remarks This property must be set to the desired value before the SOCKET_LISTEN action is taken. If the value is less than one or greater than five, it's value is silently changed to nearest legal value.

Data Type **Integer**

See Also Accept Property, Listening Property, Accept Event

Binary Property

Description Sets or returns the binary mode flag for the current socket. If set to a value of False, then subsequent reads on the socket return lines of text terminated by newlines.

Visual Basic [*form*.]*Socket*.**Binary** [= { **True** | **False** }]

Remarks This property may only be set for buffered sockets of type SOCK_STREAM, and may not be changed after a connection has been established with a remote system.

Data Type **Integer** (Boolean)

See Also BufferSize Property, RecvData Property

Blocking Property

Description Sets or returns the blocking state of the socket.

Visual Basic [*form*.]*Socket*.**Blocking** [= { **True** | **False** }]

Remarks Setting this property determines if socket operations complete synchronously or asynchronously. If set to **True**, then each socket operation (such as sending or receiving data) will return when the operation has completed or timed-out. If set to **False**, socket operations return immediately. If the operation would result in the socket blocking (such as attempting to receive data when none has been written), the error WSAEWOULDBLOCK is generated. Socket events such as Accept, Close, Read and Write are only fired if the socket is non-blocking.

If the socket is made blocking, the Blocking event is fired before the blocking operation starts, and it is possible to cancel the operation at that point.

Data Type **Integer** (Boolean)

See Also IsBlocked Property, Blocking Event

Broadcast Property

Description Determines if datagrams should be broadcast over the network

Visual Basic [*form*.]*Socket*.**Broadcast** [= { **True** | **False** }]

Remarks If set to a value of **True**, the datagram written to the socket will be broadcast to all systems on the network. Use of this property is restricted to SOCK_DGRAM socket types.

Data Type **Integer** (Boolean)

See Also InLine Property, KeepAlive Property, ReuseAddress Property, Route Property

BufferSize Property

Description Set the send and receive buffer sizes for the socket

Visual Basic [*form*.]*Socket*.**BufferSize** [= *bufsize*%]

Remarks This property sets the size of the send and receive buffers for SOCK_STREAM socket types. A buffer size of 0 indicates that no buffering should be done. Buffering reads and writes on the socket is recommended if the socket is non-blocking, and required if the **Binary** property has been set to a value of **False**.

Data Type **Integer**

See Also Binary Property, RecvData Property, Read Event

Connected Property

Description Return if the socket is connected to a remote host

Visual Basic [*form*.]*Socket*.**Connected**

Remarks This read-only property is set to a value of **True** if the socket was connected with the SOCKET_CONNECT action, or if a connection was accepted on a listening socket.

Data Type **Integer** (Boolean)

See Also Listening Property

GetFirstHost Property

Description Return the name of the first host in the host table

Visual Basic [*form*.]*Socket*.**GetFirstHost**

Remarks Reading this property returns the name of the first host in the specified host file. If there is no host file, or the file is empty, this property will return an empty string.

Data Type **String**

See Also GetNextHost Property, HostFile Property

GetNextHost Property

Description Return the name of the next host in the host table

Visual Basic [*form*.]*Socket*.**GetNextHost**

Remarks Reading this property returns the name of the next host in the specified host file. If the last host entry has been read, this property will return an empty string.

GetNextHost Property

Data Type **String**

See Also GetFirstHost Property, HostFile Property

Handle Property

Description Returns the handle (descriptor) for the current socket

Visual Basic [*form*.]*Socket*.**Handle**

Remarks This read-only property returns the handle to the current socket. If the socket has not been opened, a value of -1 is returned. This property can be used in conjunction with direct calls to the Windows Sockets API.

Data Type **Integer**

HostAddress Property

Description Set or return the IP address of the remote host

Visual Basic [*form*.]*Socket*.**HostAddress** [= *addr*$]

Remarks This property can be used to set the IP address for a remote system that you wish to communicate with. If the address is valid and matches an entry in the host table, the **HostName** property will be changed to match the address.

Data Type **String**

See Also HostName Property, LocalAddress Property, PeerAddress Property

HostFile Property

Description Sets or returns the name of the host file

Visual Basic [*form*.]*Socket*.**HostFile** [= *hostfile*$]

Remarks This property should be set to the name of the host file used by the Windows Sockets library. If no directory is provided as part of the file name, the directories listed in the PATH environment variable are searched. If the host file is located, each host entry is read into memory and returned by the **GetFirstHost** and **GetNextHost** functions.

Note that the host file must be a standard UNIX-style text file. Blank lines, or any characters that follow a hash-mark (#) are ignored.

Data Type **String**

See Also GetFirstHost Property, GetNextHost Property

HostName Property

Description Set or return the name of the remote host

Visual Basic [*form*.]*Socket*.**HostName** [= *hostname*$]

Remarks This property should be set to the name of the remote system that you wish to communicate with. If the name is found in the host table, the **HostAddress** property is updated to reflect the IP address of the host.

Note that it is legal to assign an IP address to this property, but it is not legal to assign a host name to the **HostAddress** property.

Data Type **String**

See Also HostAddress Property, LocalName Property, PeerName Property

InLine Property

Description Sets or returns if urgent data is received in-line with non-urgent data

Visual Basic [*form*.]*Socket*.**InLine** [= { **True** | **False** }]

Remarks This property controls how urgent (out-of-band) data is handled when reading data from the socket. If set to a value of **True**, urgent data is placed in the data stream along with non-urgent data. To determine if the data that is being read is urgent, the **AtMark** property can be read.

If you expect that your application must handle urgent data that is not sent in-line, it is not recommended that you use text mode or buffer the socket. Because urgent data is sent outside of the normal data stream, all receives are unbuffered.

Data Type **Integer** (Boolean)

See Also Urgent Property, Read Event

Interval Property

Description Set or return the number of milliseconds between calls to the control's Timer event

Visual Basic [*form*.]*Socket*.**Interval** [= *milliseconds*]

Remarks This property specifies the number of milliseconds between calls to the Timer event. A value of zero indicates that the timer is disabled and no events will be generated. The maximum interval value is 65536 milliseconds, which is slightly more than one minute.

There are a limited number of timers available (16 in the Windows environment). Setting the interval when no timers are available will generate a runtime error.

Data Type **Long**

See Also Timer Event

IsBlocked Property

Description Return if the current socket is blocked performing an operation

Visual Basic [*form*.]*Socket*.**IsBlocked**

Remarks This property returns **True** if the current socket is blocked performing an operation. However, since the Windows Sockets API does not allow functions to be re-entered during a blocking operation for a given *task*, this property could return **False** and a socket operation may still fail with the WSAEWOULDBLOCK error if multiple sockets have been created by the application.

Data Type **Integer** (Boolean)

See Also Blocking Property, Blocking Event

IsClosed Property

Description Return if the socket has been closed by the remote host

Visual Basic [*form*.]*Socket*.**IsClosed**

Remarks This property returns **True** if the socket connection has been closed by the remote host. Note that it is possible to continue to receive data on a closed socket if the socket was buffered.

Data Type **Integer** (Boolean)

See Also IsReadable Property, IsWritable Property

IsReadable Property

Description Return if data can be read from the socket without blocking

Visual Basic [*form*.]*Socket*.**IsReadable**

Remarks This property returns **True** if data can be read from the socket without blocking. For non-blocking sockets, this property can be checked before the application attempts to read the socket, preventing a WSAEWOULDBLOCK error.

Data Type **Integer** (Boolean)

See Also IsClosed Property, IsWritable Property, Read Event

IsWritable Property

Description Return if data can be written to the socket without blocking

Visual Basic [*form*.]*Socket*.**IsWritable**

IsWritable Property

Remarks This property returns **True** if data can be written to the socket without blocking. For non-blocking sockets, this property can be checked before the application attempts to write to the socket, preventing a WSAEWOULDBLOCK error.

Data Type **Integer** (Boolean)

See Also IsClosed Property, IsReadable Property, Write Event

KeepAlive Property

Description Set or return if "keep alives" are sent on a connected socket

Visual Basic [*form*.]*Socket*.**KeepAlive** [= { **True** | **False** }]

Remarks Setting this property to a value of **True** indicates that packets are to be sent to the remote system when no data is being exchanged to keep the connection active. This property can only be set for SOCK_STREAM socket types.

Data Type **Integer** (Boolean)

See Also Broadcast Property, InLine Property, ReuseAddress Property, Route Property

LastError Property

Description Set or return the last error that occurred on the socket

Visual Basic [*form*.]*Socket*.**LastError** [= *errval*%]

Remarks This property can be read to determine the last error that occurred for this socket. If a value is assigned to this property, it must either be zero (to clear the error) or a valid socket error code.

Data Type **Integer**

See Also Error Event

Linger Property

Description Set or return the number of seconds to linger on a close

Visual Basic [*form*.]*Socket*.**Linger** [= *nsecs*%]

Remarks Setting this property to a value greater than zero indicates that the SOCKET_CLOSE action should wait up to the specified number of seconds for any data on the socket to be written before it is closed. A value of zero indicates that the socket should be closed immediately (but gracefully, without data loss).

Data Type **Integer**

See Also Close Event

Listening Property

Description Returns if the socket is listening for connections

Visual Basic [*form*.]*Socket*.**Listening**

Remarks This read-only property returns **True** if the socket is listening for connections after the SOCKET_LISTEN action is taken.

Data Type **Integer** (Boolean)

See Also Accept Property, Backlog Property, Accept Event

LocalAddress Property

Description Return the IP address of the local host

Visual Basic [*form*.]*Socket*.**LocalAddress**

Remarks This read-only property returns the local host's IP address in dot notation (four numbers seperated by periods).

Data Type **String**

See Also HostAddress Property, LocalName Property, PeerAddress Property

LocalName Property

Description Return the name of the local host

Visual Basic [*form*.]*Socket*.**LocalName**

Remarks This read-only property returns the name of the local host. The name that is returned depends on the configuration of the TCP/IP software.

Data Type **String**

See Also HostName Property, LocalAddress Property, PeerName Property

LocalPort Property

Description Set or return the port number for a local listening socket

Visual Basic [*form*.]*Socket*.**LocalPort** [= *portno*%]

Remarks This property is used to set the port number that a local server will listen on for connections. If the port number specifies a *well-known port*, the **LocalService** property will be updated with that name.

LocalPort Property

Data Type **Integer**

See Also LocalService Property, RemotePort Property

LocalService Property

Description Set or return the name of a well-known local port

Visual Basic [*form*.]*Socket*.**LocalService** [= *servicename*$]

Remarks This property is used to set the port that a local server will listen on for connections. If the service name does not exist, an error is generated. The **LocalPort** property is updated to reflect the service's port number.

Data Type **String**

See Also LocalPort Property, RemoteService Property

Peek Property

Description Set or return if data is to be removed from the socket when read

Visual Basic [*form*.]*Socket*.**Peek** [= { **True** | **False** }]

Remarks If this property is set to a value of **True**, the data that is read from the socket is not removed, and may be read again. This property is automatically reset to a value of **False** after data has been read from the socket.

Data Type **Integer** (Boolean)

See Also RecvData Property, Read Event

PeerAddress Property

Description Return the IP address of the remote peer

Visual Basic [*form*.]*Socket*.**PeerAddress**

Remarks This read-only property returns the IP address of a the peer in dot notation (four numbers seperated by periods). The "peer" is the remote system that is either connected to, or was last sent data (if a connectionless socket is being used).

Data Type **String**

See Also HostAddress Property, LocalAddress Property, PeerName Property

PeerName Property

Description Return the name of the remote peer

Visual Basic [*form*.]*Socket*.**PeerName**

Remarks This read-only property returns the name of the peer.). The “peer” is the remote system that is either connected to, or was last sent data (if a connectionless socket is being used).

Data Type **String**

See Also HostName Property, LocalName Property, PeerAddress Property

Protocol Property

Description Set or return the protocol that should be used to create the socket

Visual Basic [*form*.]*Socket*.**Protocol** [= *proto*%]

Remarks This property may only be set before a socket has been created, or after it has been closed. Supported socket protocols are:

Value	Constant	Description
0	IPPROTO_IP	Default IP protocol. This value indicates that the protocol appropriate for the socket type should be used, and is the default.
6	IPPROTO_TCP	Transmission Control Protocol. This protocol should be used with stream sockets.
17	IPPROTO_UDP	User Datagram Protocol. This protocol should be used with datagram sockets.

There may be other protocols supported by your vendor, or in future versions of the Windows Sockets specification. Consult your TCP/IP documentation to determine what protocols are valid.

Data Type **Integer**

See Also AddressFamily Property, Type Property

RecvData Property

Description Return data read from the socket

Visual Basic [*form*.]*Socket*.**RecvData**

Remarks This property returns data that has been read from the socket, up to the number of bytes specified by the **RecvLen** property. If no data is available to be read, an error will be generated if the socket is non-blocking. If the socket is blocking, the program will stop until data is written on the socket, or the socket is closed.

RecvData Property

Note that there may be fewer bytes returned than specified by the **RecvLen** property. If the socket is in text mode, only the characters up to a newline are returned by this property.

Data Type **String**

See Also Peek Property, RecvLen Property, RecvNext Property, SendData Property, Urgent Property, Read Event

RecvLen Property

Description Set the maximum number of bytes to read, or return the number of bytes read

Visual Basic [*form*.]*Socket*.**RecvLen** [= *maxlen*%]

RecvLen Property

Remarks If set to a value, this specifies the maximum number of bytes that may be returned by the **RecvData** property. After the data has been read, **RecvLen** contains the actual number of bytes that have been read.

Note that it is common for the number of bytes to be read from the socket to be fewer than that which was specified. The application should never make any assumptions about the number of bytes received. If an error occurs, or the socket is closed, this property will have a value of zero.

Data Type **Integer**

See Also RecvData Property, RecvNext Property, Read Event

RecvNext Property

Description Return the number of bytes available to be read from the socket

Visual Basic [*form*.]*Socket*.**RecvNext**

Remarks This read-only property returns the number of bytes that are remaining in the socket and available to be read. If the socket is buffered, it is possible that this property will return a non-zero value after the socket has been closed.

Data Type **Integer**

See Also RecvData Property, RecvLen Property, Read Event

RemotePort Property

Description Set or return the port number for a remote connection

Visual Basic [*form*.]*Socket*.**RemotePort** [= *portno*%]

Remarks This property is used to set the port number that a local client will use to establish a connection with a remote system. If the port number specifies a *well-known port*, the **RemoteService** property will be updated with that name.

Data Type **Integer**

See Also LocalPort Property, RemoteService Property

RemoteService Property

Description Set or return the name of a well-known remote port

Visual Basic [*form*.]*Socket*.**RemoteService** [= *servicename*$]

Remarks This property is used to set the port that a local client will use to establish a connection with a remote system. If the service name does not exist, an error is generated. The **RemotePort** property is updated to reflect the service's port number.

Data Type **String**

See Also LocalService Property, RemotePort Property

ReuseAddress Property

Description Set or return if an address can be reused

Visual Basic [*form*.]*Socket*.**ReuseAddress** [= { **True** | **False** }]

Remarks Setting this property to a value of **True** allows the address that the socket is listening on to be reused. If the **LocalPort** property is set to 0, then this property is ignored.

Data Type **Integer** (Boolean)

See Also Broadcast Property, InLine Property, KeepAlive Property, Route Property

Route Property

Description Set or return if packets should be routed

Visual Basic [*form*.]*Socket*.**Route** [= { **True** | **False** }]

Remarks Setting this property to **False** tells the socket library that packets are not to be routed, but rather sent directly to the network interface.

Route Property

Data Type **Integer** (Boolean)

See Also Broadcast Property, InLine Property, KeepAlive Property, ReuseAddress Property

SendData Property

Description Write data to the socket

Visual Basic [*form*.]*Socket*.**SendData** = *data*$

Remarks By assigning a value to this property, the data in the string is written to the socket. If the socket is buffered, the data is copied to the send buffer and control immediately returns to the program. If the socket is non-blocking, and the socket is out of buffer space, the error WSAEWOULDBLOCK will be generated. If the socket is blocking, the program will wait until the data can be sent.

The maximum number of bytes written on the socket is determined by the **SendLen** property. It is possible that fewer than the number of bytes specified will be written to the socket. After the data has been written, the **SendLen** property will be changed to reflect the number of bytes actually written to the socket. If the socket is non-blocking and the send fails with WSAEWOULDBLOCK, the **Write** event will be fired when the socket can be written to again.

Data Type **String**

See Also SendLen Property, Timeout Property, Urgent Property, Timeout Event, Write Event

SendLen Property

Description Set or return the maximum number of bytes to write to the socket

Visual Basic [*form*.]*Socket*.**SendLen** [= *datalen*%]

Remarks If set to a value, this property specifies the maximum number of bytes that may be written to the socket. If this value is greater than the length of the string being sent, the value is ignored and all of the characters are sent.

After data has been written to the socket, this property is updated to reflect the actual number of bytes written.

Data Type **Integer**

See Also SendData Property, Write Event

Shutdown Property

Description Stop reading and/or writing on the socket

Visual Basic [*form*.]*Socket*.**Shutdown** [= *what*%]

Remarks This write-only property shuts down reading and/or writing on the socket. Any further attempt to send or receive data will return an error on the socket. The possible values that may be assigned to this property are:

Value	Constant	Description
0	SOCKET_READ	All subsequent attempts to read data from the socket are disallowed. If the socket is buffered, it is possible that data may still be read until the buffer is exhausted.
1	SOCKET_WRITE	All subsequent attempts to write data to the socket are disallowed. If the socket is buffered, it is possible that data may be written until the buffer is full.
2	SOCKET_READWRITE	Both reads and writes to the socket are disallowed.

Note that shutting down a socket is not the same as closing it. The socket will remain connected, and no resources will be freed until the SOCKET_CLOSE action is taken.

Data Type **Integer**

See Also IsReadable Property, IsWritable Property

State Property

Description Return the current state of the socket

Visual Basic [*form*.]*Socket*.**State**

Remarks This read-only property returns the state of the socket. This property should be checked on blocking sockets to determine if the socket is in use before taking some action. The possible values returned by this property are:

Value	Constant	Description
0	SOCKET_UNUSED	The socket has not been created. Attempts to use the socket will generate an error.
1	SOCKET_IDLE	The socket exists, but is not currently in use. A blocking socket operation can be executed at this point.
2	SOCKET_LISTENING	The socket is listening for connections from remote hosts
3	SOCKET_CONNECTION	The socket is in the process of connecting to a remote host
4	SOCKET_RECEIVING	The socket is in the process of receiving data
5	SOCKET_SENDING	The socket is in the process of sending data

State Property

6	SOCKET_CLOSING	The socket is being closed. Subsequent attempts to access the socket will result in an error.

Note that for non-blocking sockets, the only possible states that may be returned are SOCKET_UNUSED, SOCKET_IDLE or SOCKET_CLOSING.

Data Type **Integer**

See Also IsBlocked Property, Blocking Event, Cancel Event, Timeout Event

Timeout Property

Description Set or return the amount of time until a blocking operation fails

Visual Basic [*form*.]*Socket*.**Timeout** [= *msecs*&]

Remarks Setting this property specifies the number of milliseconds until a blocking operation fails with the error WSAETIMEDOUT. A value of zero indicates that the blocking operation should wait indefinitely.

Data Type **Long**

See Also Accept Property, Action Property, RecvData Property, SendData Property, Timeout Event

Type Property

Description Sets or returns the type of socket that should be created

Visual Basic [*form*.]*Socket*.**Type** [= *socktype*%]

Remarks This property may only be set before a socket has been created, or after it has been closed. Supported socket types are:

Value	Constant	Description
1	SOCK_STREAM	Stream socket used with the TCP protocol. This type of socket provides a reliable byte stream that may be read similar to the way a file is read.
2	SOCK_DGRAM	Datagram socket used with the UDP protocol. This type of socket is used to transfer datagrams using a fast (but unreliable) protocol. Datagrams may arrive out of sequence, or not at all. Retransmission of lost datagrams is the responsibility of the application.
3	SOCK_RAW	A raw socket that is commonly used to send ICMP messages over the network. This socket type may or may not be supported by your TCP/IP vendor.

Type Property

There may be other types of sockets supported by your vendor, or in future versions of the Windows Sockets specification. Consult your TCP/IP documentation to determine what socket types are valid.

Data Type **Integer**

See Also AddressFamily Property, Protocol Property

Urgent Property

Description Send or receive urgent data

Visual Basic [*form*.]*Socket*.**Urgent** [= { **True** | **False** }]

Remarks This Boolean property affects how the **RecvData** and **SendData** properties read or write data to the socket. If set to a value of **True**, urgent (out-of-band) data will be read or written. All reads or writes of urgent data are unbuffered. The property value will automatically be reset to a value of **False** after the socket has been read or written. Not all implementations may support more than one byte of urgent data if the data is not being received in-line. Refer to the **InLine** property for additional information.

Data Type **Integer** (Boolean)

See Also RecvData Property, SendData Property, Read Event

Vendor Property

Description Returns the name of the TCP/IP vendor

Visual Basic [*form*.]*Socket*.**Vendor**

Remarks This read-only property returns a string that contains the name of the vendor that has supplied the Windows Sockets library that is being used.

Data Type **String**

See Also Version Property

Version Property

Description Returns the version of the Windows Sockets library

Visual Basic [*form*.]*Socket*.**Version**

Remarks This read-only property returns a string that specifies the version of the Windows Sockets library that is being used. Note that at least version 1.1 is required for use with this control.

Data Type **String**
See Also Vendor Property

Accept Event

Description The Accept event is generated when a remote host connects to a listening socket.

Visual Basic **Sub** *Socket*_**Accept (**[*Index* **As Integer**,] *SocketID* **As Integer)**

Remarks This event is generated for sockets that are listening for connections from a remote host. A connection with the remote system is not actually established until it has been accepted by the listening server.

The *SocketID* argument specifies the socket descriptor of the listening socket. To accept the connection, one of the following actions must be taken:

- The program sets the **Action** property to a value of SOCKET_ACCEPT. This allows the socket that was listening to connect with the remote host. However, this method closes the listening socket, so only one host can establish a connection with the application.

- A second socket has it's **Accept** property set to the value of *SocketID*. The second socket accepts the connection on behalf of the server, and the original socket may continue to listen for additional connections.

Once the connection has been established, the **PeerAddress** or **PeerName** properties may be used to determine the name of the remote host that has connected with the application.

See Also Accept Property, Action Property, PeerAddress Property, PeerName Property

Blocking Event

Description The Blocking event is generated whenever a blocking operation occurs

Visual Basic **Sub** *Socket*_**Blocking (**[*Index* **As Integer**,] *Status* **As Integer,** *Cancel* **As Integer)**

Remarks This event is generated immediately before a blocking operation takes place, and provides a kind of blocking hook function for the application.

The *Status* argument specifies which blocking action is about to be taken. For a list of values, refer to the **State** property.

The *Cancel* argument allows the blocking operation to be canceled if the value is set to **True**. The blocking function will fail with the error WSAEINTR.

It is not recommended that code be placed in this event that would take any significant amount of time to complete or calls **DoEvents** to yield time to other tasks.

See Also State Property, Cancel Event

Cancel Event

Description The Cancel event is generated when a blocking operation is canceled

Visual Basic **Sub** *Socket*_**Cancel (**[*Index* **As Integer**,] *Status* **As Integer,** *Response* **As Integer)**

Remarks This event is generated when a blocking operation on the socket, such as sending or receiving data, is canceled with the SOCKET_CANCEL action.

The *Status* argument specifies which blocking operation was canceled. For a list of values, refer to the **State** property.

The *Response* argument determines if a WSAEINTR error is generated in response to the canceled operation. The possible values are:

Value	Constant	Description
0	SOCKET_ERRIGNORE	Ignore the canceled event and return as though the operation completed successfully.
1	SOCKET_ERRDISPLAY	Return the WSAEINTR error to the canceled socket operation. This is the default response to the event.

Setting *Response* to ignore the canceled event can result in unexpected errors. For example, if a blocking read on the socket is cancelled, the WSAEWOULDBLOCK error may be returned.

See Also State Property, Blocking Event, Error Event

Close Event

Description The Close event is generated when the socket is closed

Visual Basic **Sub** *Socket*_**Close (** [*Index* **As Integer**] **)**

Remarks This event occurs when a remote host closes the socket connection. It is not generated when the SOCKET_CLOSE action is taken. When this event is generated, the local socket should be closed to free the resources allocated to it.

See Also Action Property, Connect Event

Connect Event

Description The Connect event is generated when a connection is established

Visual Basic **Sub** *Socket*_**Connect (** [*Index* **As Integer**] **)**

Remarks This event is generated when a connection is made with a remote host as a result of a SOCKET_CONNECT action, or when a connection is accepted on a listening socket.

See Also Action Property, Close Event

Error Event

Description The Error event is generated when a socket error occurs

Visual Basic **Sub** *Socket*_**Error (**[*Index* **As Integer**,] *ErrCode* **As Integer,** *ErrMsg* **As String,** *Response* **As Integer)**

Remarks This event is generated when an error occurs during a socket operation. Visual Basic errors (e.g.: Invalid Property) will not generate this event.

The *ErrCode* argument specifies the error that has occurred on the socket. For a list of errors, refer to Appendix A.

The *ErrMsg* argument is a string that describes the error that occurred.

The *Response* argument determines if the socket error generates a Visual Basic error. The possible values are:

Value	Constant	Description
0	SOCKET_ERRIGNORE	Ignore the error and return as though the operation completed successfully.
1	SOCKET_ERRDISPLAY	Return the error to Visual Basic, which may be trapped by the application. This is the default response to the event.

See Also Cancel Event

Read Event

Description The Read event is generated when data is available to be read

Visual Basic **Sub** *Socket*_**Read (**[*Index* **As Integer**,] *DataLength* **As Integer,** *IsUrgent* **As Integer)**

Remarks This event is generated for non-blocking sockets when data is available to be read from the socket.

The *DataLength* argument specifies the number of bytes that can be read from the socket.

The *IsUrgent* argument specifies if the data to be read is marked as urgent. A non-zero value indicates that the **Urgent** property should be set to **True** to receive the out-of-band data. This flag is only set if the **InLine** property is **False.**

It is possible for the **RecvNext** property to return a value greater than *DataLength* if additional data has been written to the socket while executing code inside the event.

See Also IsReadable Property, Write Event

Timeout Event

Description The Timeout event is fired when a blocking operation times out

Visual Basic **Sub** *Socket*_**Timeout (**[*Index* **As Integer**,] *Status* **As Integer,** *Response* **As Integer)**

Remarks This event is generated when a blocking socket operation, such as sending or receiving data, times out.

The *Status* argument specifies which blocking operation timed out. For a list of values, refer to the **State** property.

The *Response* argument determines if a WSAETIMEDOUT error is generated. The possible values are:

Value	Constant	Description
0	SOCKET_ERRIGNORE	Ignore the timeout and return as though the operation completed successfully.
1	SOCKET_ERRDISPLAY	Return the WSAETIMEDOUT error to the timed-out socket operation. This is the default response to the event.

See Also State Property, Timeout Property, Cancel Event

Timer Event

Description The Timer event is fired when the control's preset timer interval expires

Visual Basic **Sub** *Socket*_**Timer (** [*Index* **As Integer**] **)**

Remarks This event is generated when the control's timer interval has elapsed. The frequency is specified in milliseconds by setting the Interval property.

See Also Interval Property

Write Event

Description The Write event is generated when data can be written to the socket

Visual Basic **Sub** *Socket*_**Write (** [*Index* **As Integer**] **)**

Remarks This event is generated for non-blocking sockets when data can be written to the socket after a previous attempt failed with the WSAEWOULDBLOCK error.

See Also IsWritable Property, Read Event

Appendix A - Error Codes

The error codes listed here are based on Windows Sockets errors (which, in turn, are based on the error codes that are returned by the Berkeley sockets implementation). The base error value has been increased by 14000 to be compatible with Visual Basic.

24004	WSAEINTR	Blocking function was canceled
24009	WSAEBADF	Invalid socket descriptor passed to function
24013	WSAEACCES	Access denied
24014	WSAEFAULT	Invalid address passed to function
24022	WSAEINVAL	Invalid socket function call
24024	WSAEMFILE	No socket descriptors are available
24035	WSAEWOULDBLOCK	Socket would block on this operation
24036	WSAEINPROGRESS	Blocking function in progress
24037	WSAEALREADY	Function being canceled has already completed
24038	WSAENOTSOCK	Invalid socket descriptor passed to function
24039	WSAEDESTADDRREQ	Destination address is required
24040	WSAEMSGSIZE	Datagram was too large to fit in specified buffer
24041	WSAEPROTOTYPE	Specified protocol is the wrong type for this socket
24042	WSAENOPROTOOPT	Socket option is unknown or unsupported
24043	WSAEPROTONOSUPPORT	Specified protocol is not supported
24044	WSAESOCKTNOSUPPORT	Specified socket type is not supported in this address family
24045	WSAEOPNOTSUPP	Socket operation is not supported
24046	WSAEPFNOSUPPORT	Specified protocol family is not supported
24047	WSAEAFNOSUPPORT	Specified address family is not supported by this protocol
24048	WSAEADDRINUSE	Specified address is already in use
24049	WSAEADDRNOTAVAIL	Specified address is not available
24050	WSAENETDOWN	Network subsystem has failed
24051	WSAENETUNREACH	Network cannot be reached from this host
24052	WSAENETRESET	Network dropped connection on reset
24053	WSAECONNABORTED	Connection was aborted due to timeout or other failure
24054	WSAECONNRESET	Connection was reset by remote network
24055	WSAENOBUFS	No buffer space is available
24056	WSAEISCONN	Socket is already connected
24057	WSAENOTCONN	Socket is not connected
24058	WSAESHUTDOWN	Socket connection has been shut down
24060	WSAETIMEDOUT	Operation timed out before completion
24061	WSAECONNREFUSED	Connection refused by remote network
24064	WSAEHOSTDOWN	Remote host is down
24065	WSAEHOSTUNREACH	Remote host is unreachable
24091	WSASYSNOTREADY	Network subsystem is not ready for communication
24092	WSAVERNOTSUPPORTED	Requested version is not available
24093	WSANOTINITIALIZED	Windows sockets library not initialized
25001	WSAHOST_NOT_FOUND	Authoritative Answer Host not found
25002	WSATRY_AGAIN	Non-authoritative Answer Host not found
25003	WSANO_RECOVERY	Non-recoverable error
25004	WSANO_DATA	No data record of requested type