NETMANAGE

WinSock TCP ActiveX Control

The following Help Topics are available:

For Help on Help, Press F1

Net*Manage*

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

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

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

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

General

The WinSock TCP ActiveX Control implements the WinSock Transmission Control Protocol (TCP) for both client and server applications.

Invisible to the user, the TCP Control provides easy access to TCP network services. It can be used by both Delphi and C++ programmers. To write client or server applications you do not need to understand the details of TCP or to call low level WinSock APIs. By setting properties and calling methods on the control, you can easily connect to a remote machine and exchange data in both directions. Events are used to notify you of network activities.

Properties

Following is an alphabetical list of all properties supported by the TCP Control.

Note: Some common ActiveX properties of the control, such as Name, Index, About Box, and others, may appear in the Object Browser but are not documented here.

BytesReceived

LocalHostName

LocalIP

LocalPort

RemoteHost

RemoteHostIP

RemotePort

SocketHandle

State

Naming Conventions

Objects described in the Properties, Methods and Events section are preceded by the required parameter: object.   During execution object translates to the name of the control. The actual object name will be:

`NMTCP`*`n`*

where *n* is the number identifier.   For example, the first TCP in a form becomes TCP1, the second is TCP2 and so forth.

BytesReceived

Description

Advanced property.   It shows the amount of data received (currently in the receive buffer).   The GetData method should be used to retrieve data.

Syntax

object.**BytesReceived**

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

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

Long.

Default Value

0.

Range

0 - 0xFFFFFFFF

LocalHostName

Description

Local machine name.

Syntax

object.**LocalHostName**

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

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A

LocalIP

Description

The IP address of the local machine.   It has the format:

```
    number.number.number.number
```

Syntax

*object.***LocalIP**

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

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A

LocalPort

Description

For the client, this designates the local port to use. Specify port 0 if the application does not need a specific port. In this case, the control will select a random port. After a connection is established, this is the local port used for the TCP connection.

For the server, this is the local port to listen on. If port 0 is specified, a random port is used. After calling the Listen method, the property contains the actual port that has been selected.

Syntax

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

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

Permission

W (Read/Write).

Availability

D (Design).

Data Type

Long.

Default Value

0.

Range

0 - 65535

RemoteHost

Description

The remote machine to connect to if the RemoteHost parameter of the Connect method is not specified.   You can either provide a host name or an IP address string in dotted format.

Syntax

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

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

Permission

W (Read/Write).

Availability

D (Design).

Data Type

String.

Default Value

Empty.

Range

N/A

Comment

This is the default property of the control.

RemoteHostIP

Description

For the client, after a connection has been established (i.e., after the Connect event has been activated), this property contains the IP string of the remote machine in dotted format.

For server, after an incoming connection request (ConnectionRequest event), this property contains the IP string (in dotted format) of the remote machine initiating the connection.

Syntax

*object.***RemoteHostIP**

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

Permission

R (Read-only).

Availability

R (Runtime).

Data Type

String.

Default Value

Empty.

Range

N/A

RemotePort

Description

For the client, this is the remote port number to which to connect if the RemotePort parameter of the Connect method is not specified.

For the server, after an incoming connection request event, (ConnectionRequest has been activated) this property contains the port that the remote machine uses to connect to this server.

Syntax

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

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

Permission

W (Read/Write).

Availability

D (Design).

Data Type

Long.

Default Value

0.

Range

0 - 65535

SocketHandle

Description

This is the socket handle the control uses to communicate with the WinSock layer.

Syntax

*object.***SocketHandle**

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

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Long.

Default Value

-1.

Range

-1 - 65535

Comment

This property is for advanced programmers. You can use SocketHandle in direct WinSock API calls. However, you should be aware that if WinSock calls are used directly, certain events may not be activated appropriately.

State

Description

The state of the control, expressed as an enum type.

Syntax

*object.***State**

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

Permission

R (Read only).

Availability

R (Runtime).

Data Type

Integer.

Default Value

0.

Range

0-9. Constants defined for enum types in this property are:

Enum Type	Meaning
sckClosed = 0	Closed
sckOpen = 1	Open
sckListening = 2	Listening
sckConnectionPending = 3	Connection pending
sckResolvingHost = 4	Resolving host
sckHostResolved = 5	Host resolved
sckConnecting = 6	Connecting
sckConnected = 7	Connected
sckClosing = 8	Peer is closing the connection
sckError = 9	Error

Property Page

One property page is provided for viewing and editing the following properties:

- RemoteHost
- RemotePort
- LocalPort

Methods

The methods performed by the TCP Control are:

Accept

Close

Connect

GetData

Listen

PeekData

SendData

Accept

See Also

Description

This method is used to accept an incoming connection when handling a ConnectionRequest event.

Return Value

Void.

Syntax

object.**Accept** *RequestID*

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

Parameters

RequestID

> The incoming connection request identifier. This should be the requestID passed in the ConnectionRequest event.
> Data Type: Long
> Param: IN

Comment

Accept should be used on a new control instance (other than the one that is in the listening state.)

ConnectionRequest Event

Close

Description

Closes a TCP connection or a listening socket for both client and server.

Return Value

Void.

Syntax

*object.***Close**

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

Parameters

None.

Connect

Description

Initiates connection to remote machine.

Return Value

Void.

Syntax

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

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

Parameters

RemoteHost

> Optional. If this parameter is missing, Connect will connect to the remote host specified in the RemoteHost property. If this parameter is missing, Connect will connect to the remote host specified in the RemoteHost property.
>
> Data Type: VARIANT
> Param: IN

RemotePort

> Optional. If this parameter is missing, Connect will connect to the remote port specified in the RemotePort property. If this parameter is missing, Connect will connect to the remote port specified in the RemotePort property.
>
> Data Type: VARIANT
> Param: IN

Comment

If the connection is successfully established, the Connect event will be activated. If an error occurs during connection, the Error event will be activated.

GetData

Description

Retrieves data.

Return Value

Void.

Syntax

object.**GetData** *data* [,*type*] [,*maxLen*]

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

Parameters

Data

Stores retrieved data after the method returns successfully. If there is not enough data available for requested type, *data* will be set to Empty.
Data Type: VARIANT
Param: OUT

Type

Optional. Type of data to be retrieved.
Data Type: VARIANT
Param: IN
Default Value: vbArray + vbByte
Currently, the following variant types are supported.

Type	C++	VB Type
Byte	VT_UI1	vbByte
Integer	VT_I2	vbInteger
Long	VT_I4	vbLong
Single	VT_R4	vbSingle
Double	VT_R8	vbDouble
Currency	VT_CY	vbCurrency
Date	VT_DATE	vbDate
Boolean	VT_BOOL	vbBoolean
SCODE	VT_ERROR	vbError
String	VT_String	vbString
Byte Array	VT_ARRAY | VT_UI1	vbArray + vbByte

maxLen

Optional length parameter. This parameter can be used to specify the desired size when receiving a byte array or a string . If this parameter is missing for byte array or string, all available data will be retrieved. If provided, for data types other than byte array and string, this parameter is ignored.
Data Type: VARIANT
Param: IN
Default Value: All data available.

Comments

If the type is specified as vbString, string data is converted to UNICODE before returning to the user.

Listen

Description

It includes creating a socket and putting the socket in the listening mode.

Return Value

Void.

Syntax

*object.***Listen**

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

Parameters

None.

Comment

When there is an incoming connection, the ConnectionRequest event is activated.   When handling ConnectionRequest, the application should use the Accept method (on a new control instance) to accept the connection.

PeekData

Description

Similar to GetData except PeekData does not remove data from input queue.

Return Value

Void.

Syntax

object.**PeekData** *data*, [*type*,] [*maxLen*]

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

Parameters

Data

Stores retrieved data after the method returns successfully. If there is not enough data available for requested type, *data* will be set to Empty.
Data Type: VARIANT
Param: OUT

Type

Optional. Type of data to be retrieved.
Data Type: VARIANT
Param: IN
Default Value: vbArray + vbByte
Currently, the following variant types are supported.

Type	**C++**	**VB Type**
Byte	VT_UI1	vbByte
Integer	VT_I2	vbInteger
Long	VT_I4	vbLong
Single	VT_R4	vbSingle
Double	VT_R8	vbDouble
Currency	VT_CY	vbCurrency
Date	VT_DATE	vbDate
Boolean	VT_BOOL	vbBoolean
SCODE	VT_ERROR	vbError
String	VT_String	vbString
Byte Array	VT_ARRAY | VT_UI1	vbArray + vbByte

maxLen

Optional length parameter. This parameter can be used to specify the desired size when receiving a byte array or a string . If this parameter is missing for byte array or string, all available data will be retrieved. If provided, for data types other than byte array and string, this parameter is ignored.
Data Type: VARIANT
Param: IN
Default Value: All data available.

Comments

If the type is specified as vbString, string data is converted to UNICODE before returning to the user.

SendData

Description

Sends data to peer.

Return Value

Void.

Syntax

*object.***SendData** *data*

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

Parameters

Data

Data to be sent. For binary data, byte array should be used.
Data Type: VARIANT
Param: IN
Currently, the following variant types are supported.

Type	**C++**	**VB Type**
Byte	VT_UI1	vbByte
Integer	VT_I2	vbInteger
Long	VT_I4	vbLong
Single	VT_R4	vbSingle
Double	VT_R8	vbDouble
Currency	VT_CY	vbCurrency
Date	VT_DATE	vbDate
Boolean	VT_BOOL	vbBoolean
SCODE	VT_ERROR	vbError
String	VT_String	vbString
Byte Array	VT_ARRAY | VT_UI1	vbArray + vbByte

Comments

When a UNICODE string is passed in, it is converted to an ANSI string before being sent out on the network.

Events

The list of events follows.

Close

Connect

ConnectionRequest

DataArrival

Error

SendComplete

SendProgress

Error

Description

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

Syntax

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

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

Parameters

ErrCode

An integer that defines the error code. For a list of possible WinSock error codes see WinSock Error Codes.

Description

String containing error information.

Scode

The long SCODE.

Source

String describing the error source.

HelpFile

String containing help file name.

HelpContext

Help file context.

CancelDisplay

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

Close

Description

The event is activated when the peer closes the connection.   Applications should use the Close method to correctly close the connection.

Syntax

*object*_**Close**

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

Parameters

None.

Connect

Description

The event is activated when a connection has been successfully established.   After this event is activated, you can send or receive data on the control.

Syntax

*object*_**Connect**

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

Parameters

None.

ConnectionRequest

Description

The event is activated when there is an incoming connection request. RemoteHostIP and RemotePort properties store the information about the client after the event is activated.

The server can decide whether or not to accept the connection. If the incoming connection is not accepted, the peer (client) will get the Close event. Use the Accept method (on a new control instance) to accept an incoming connection.

Syntax

*object*_**ConnectionRequest** (*RequestID* **As Long***)*

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

Parameters

RequestID

The incoming connection request identifier. This parameter should be passed to the Accept method on the second control instance.
Data Type: Long
Param: IN

DataArrival

Description

The event is activated when new data arrives.

Syntax

*object*_**DataArrival** (*BytesTotal* **As Long***)*

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

Parameters

BytesTotal

The total amount of data that can be retrieved.
Data Type: Long
Param: IN

Comment

This event will not be activated if you do not retrieve all the data in one GetData call. It is activated only when there is new data. You can always use the BytesReceived property to check how much data is available at any time.

SendComplete

Description

The event is activated when the send buffer is empty.

Syntax

*object*_**SendComplete**

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

Parameters

None.

SendProgress

Description

This event notifies the user of sending progress.   It is activated when more data has been accepted by the stack.

Syntax

*object*_**SendProgress** (*BytesSent* **As Long**, *BytesRemain* **As Long**)

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

Parameters

BytesSent

The number of bytes that have been sent since the last time this event was activated.
Data Type: Long
Param: IN

BytesRemain

The number of bytes in the send buffer waiting to be sent.
Data Type: Long
Param: IN

Localization

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

Sample Session

The sample session illustrates a real life scenario using the TCP ActiveXControl.   From this example you can see how to write for both the client and server.

TCP Client

TCP Server

TCP Client

A TCP client actively initiates a connection to a remote machine. You would then call the Connect method.

When the connection has been established successfully, a Connect event occurs. If the remote host rejected the connection, an Error event occurs.

After a connection has been established, use the SendData method to stream data to a remote machine. A DataArrival event occurs when there is incoming data. Use the Close method to terminate the connection.

TCP Server

A TCP server listens at a particular port for incoming connections. To write an echo server which echoes back all data it receives, the server would listen at the standard echo port 7. You should set LocalPort to 7 and call the Listen method. When there is an incoming connection request, a ConnectionRequest event occurs. Use another instance of TCP Control to accept the incoming connection. When this is complete, the application can send and receive data on the newly accepted connection as described in TCP Client section.

ActiveX

WinSock Error Codes

The following error codes apply to the WinSock ActiveX Controls.

Error Number	Error Message
10004	The operation is canceled
10013	The requested address is a broadcast address, but flag is not set
10014	Invalid argument
10022	Socket not bound, invalid address or listen is not invoked prior to accept
10024	No more file descriptors are available, accept queue is empty
10035	Socket is non-blocking and the specified operation will block
10036	A blocking Winsock operation is in progress
10037	The operation is completed. No blocking operation is in progress.
10038	The descriptor is not a socket
10039	Destination address is required
10040	The datagram is too large to fit into the buffer and is truncated
10041	The specified port is the wrong type for this socket
10042	Option unknown, or unsupported
10043	The specified port is not supported
10044	Socket type not supported in this address family
10045	Socket is not a type that supports connection oriented service
10047	Address Family is not supported
10048	Address in use
10049	Address is not available from the local machine
10050	Network subsystem failed
10051	The network cannot be reached from this host at this time
10052	Connection has timed out when SO_KEEPALIVE is set
10053	Connection is aborted due to

	timeout or other failure
10054	The connection is reset by remote side
10055	No buffer space is available
10056	Socket is already connected
10057	Socket is not connected
10058	Socket has been shut down
10060	The attempt to connect timed out
10061	Connection is forcefully rejected
10201	Socket already created for this object
10202	Socket has not been created for this object
11001	Authoritative answer: Host not found
11002	Non-Authoritative answer: Host not found
11003	Non-recoverable errors
11004	Valid name, no data record of requested type