

Telnet ActiveX Control

**Telnet Services
ActiveX Control
for Microsoft® Windows™**

Version 5.2

**Copyright © 1995 - 2003 by Distinct Corporation
All rights reserved**

Distinct Corporation

3315 Almaden Expressway
San Jose, CA 95118 USA

Phone: +1 408-445-3270

Fax: +1 408-445-3274

Email: sales@distinct.com

WWW: <http://www.distinct.com>

Disclaimer

Distinct Corporation makes no warranties as to the contents of this documentation and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Information in this manual is subject to change without notice and does not represent a commitment on the part of Distinct Corporation. The Software described in this manual is furnished under a license agreement and may be used or copied only in accordance with the terms of that agreement.

Copyright Notice

© 1995 - 2003 by Distinct Corporation. All rights reserved.

No part of this publication may be reproduced, transmitted or translated into any language by any means without the express written permission of Distinct Corporation.

Trademarks

Distinct is a registered trademark and Visual Internet Toolkit and Distinct Telnet are trademarks of Distinct Corporation. Windows is a registered trademark of Microsoft Corporation. Other product names are trademarks or registered trademarks of their respective owners.

Last updated June 5th, 2003

Published in the United States of America

Table of Contents

Table of Contents.....	1
1 Overview of the Distinct Telnet Component.....	3
1.1 Introduction.....	3
1.2 Usage of the Distinct Telnet Component.....	3
1.3 Telnet Property Summary.....	4
1.4 Telnet Event Summary.....	4
1.5 Telnet Method Summary.....	5
1.6 D_TNET.TXT.....	5
2 Properties for Telnet.....	7
2.1 Action.....	7
2.2 BinaryData.....	8
2.3 Echo.....	9
2.4 FirewallPort.....	10
2.5 FirewallServer.....	11
2.6 FwAddrType.....	12
2.7 FwAuthMethods.....	13
2.8 FwPassword.....	14
2.9 FwSocksVer.....	15
2.10 FwUsername.....	16
2.11 HostName.....	17
2.12 Port.....	18
2.13 Receive.....	19
2.14 ReceiveCount.....	20
2.15 ReceiveLen.....	21
2.16 Result.....	22
2.17 Send.....	23
2.18 TerminalType.....	24
3 Events for Telnet.....	25
3.1 OnClose.....	25
3.2 OnConnect.....	26
3.3 OnError.....	27
3.4 OnReceive.....	28
4 Methods for Telnet.....	29
4.1 Abort.....	29
4.2 Connect.....	30
4.3 Disconnect.....	31
4.4 FwConnect.....	32
4.5 ReceiveB.....	34
4.6 SendB.....	36
Registry Entries.....	37
Telnet Client Registry Entries.....	37

1 Overview of the Distinct Telnet Component

1.1 Introduction

The Distinct Telnet ActiveX control allows you to integrate Telnet based login capabilities into their applications. Telnet provides the ability to have a basic connection. It is the protocol used for all terminal emulation over TCP/IP. Firewall proxy support is embedded into the Telnet ActiveX control so that a Telnet session can be established via a firewall. Distinct's Telnet ActiveX control encapsulates the establishment of the TCP connection and handles the Telnet option negotiations. This means that only the application's data is sent and received over the Telnet connection - this frees the application from generating or filtering out escape sequences which control the session parameters.

1.2 Usage of the Distinct Telnet Component

See the section entitled "Using Distinct ActiveX controls in various environments" for details on how to add the control to your project.

After placing a Telnet ActiveX control into a form, some properties can be set at design time. Although property settings default to their most common value, some properties may need to be changed at design time. The Echo property defaults to remote echo, the Port property defaults to server port 23 and the TerminalType property defaults to "tty". These three properties should be preset at design time, although they can also be changed during run time unless a Telnet session is connected. Please check the reference pages of these properties for more details.

The HostName property is usually set at run time right before a session is established. This allows the application to request the host name or internet address of the Telnet server from the user. If an application will always connect to the same host, then the HostName property can also be set at design time to minimize user interaction.

The ReceiveLen property controls the maximum number of bytes of data that are copied every time the Receive property is accessed. The preset value of 100 should be sufficient for most applications, but any value greater than 0 can be used. Setting this property to 1 will cause the application to read just one byte of data at a time. This property can be modified at any time even while connected.

The remaining properties (Action, Receive, ReceiveCount and Send) can only be accessed at run time. The Receive, ReceiveCount and Send properties can only be used while a Telnet session is connected. To establish a Telnet session call the Connect method(or set the value of the Action property to ACTION_CONNECT). If the connection can be established, the OnConnect event will occur before the next line of code is reached. At this point, the Send and Receive properties can be accessed to transfer data to and from the Telnet server. The ReceiveB and SendB methods can be used to transfer binary data to and from the Telnet server.

For some users, the local machine is located on a different subnet than the remote host and the only way of communication is through a firewall. If this is the case, then the built-in firewall support of the Telnet ActiveX control can be used to establish a Telnet session via a firewall by setting the Action property to ACTION_FW_CONNECT (or by calling the FwConnect method). In addition to setting the properties for a regular Telnet session, the FirewallServer, FirewallPort, FwAddrType, FwAuthMethods, FwUsername and FwPassword properties must also be set before setting the Action property to ACTION_FW_CONNECT. If the connection can be established, the OnConnect event will occur before the next line of code is reached.

The Result property contains the number of bytes sent to the remote machine or received from the remote machine. The value of this property can be checked after each send operation to ensure that all of the data was sent and after each receive operation to determine the number of bytes read from the receive buffer.

Once a connection is no longer needed, the Action property must be set to ACTION_DISCONNECT (or the Disconnect method must be called). After the session is disconnected, the OnClose event will occur before the next line of code is reached. Applications must disconnect all connected sessions before terminating.

1.3 Telnet Property Summary

Action

Connect, disconnect or abort a Telnet session

BinaryData

Contains the binary data received following a call to the RetrieveB method.

Echo

Set remote or local echo

FirewallPort

Firewall server port

FirewallServer

Name or dotted decimal internet address of firewall server

FwAddrType

The address type of the destination host

FwAuthMethods

Firewall authentication methods

FwUsername

Firewall username

FwPassword

Firewall password

FwSocksVer

The firewall server version

HostName

Name of host or dotted decimal internet address

Port

Telnet service port on host

Receive

Receive buffer

ReceiveCount

Number of bytes waiting in receive buffer

ReceiveLen

Maximum number of bytes to read from receive buffer

Result

Number of bytes sent or received

Send

Send buffer

TerminalType

Type of terminal emulated by the application

1.4 Telnet Event Summary

OnClose

Telnet connection has been closed

OnConnect

Telnet connection has been established

OnError

Local error has occurred

OnReceive

More data has been received

1.5 Telnet Method Summary

Abort

Abort a remote session

Connect

Connect to remote server

Disconnect

Close connection to remote server

FwConnect

Connect to remote server via a firewall

ReceiveB

Receive binary data

SendB

Send binary data

1.6 D_TNET.TXT

The following provides a complete listing of the D_TNET.TXT definition file. If your application uses more than one Distinct ActiveX control in the same form, then some definitions will conflict. For example, the FTP Client ActiveX control includes the definition

```
Global Const ACTION_DISCONNECT = 3
```

in the D_FTP.TXT file and the Telnet ActiveX control includes the definition

```
Global Const ACTION_DISCONNECT = 2
```

in the D_TNET.TXT file. To avoid this conflict, you must rename at least one of the constants (for example, FTP_ACTION_DISCONNECT or TNET_ACTION_DISCONNECT).

```
' Telnet ActiveX Control
' (C) Copyright 1995 - 1998 Distinct Corporation
' All rights reserved

' actions
Global Const ACTION_NONE = 0
Global Const ACTION_CONNECT = 1
Global Const ACTION_DISCONNECT = 2
Global Const ACTION_ABORT = 3
Global Const ACTION_FW_CONNECT = 4

' echo modes
Global Const ECHO_REMOTE = 0
Global Const ECHO_LOCAL = 1

' the address type of the destination host
Global Const FW_ADDR_IP4 = 1
Global Const FW_ADDR_DNS = 3
Global Const FW_ADDR_IP6 = 4
```

```
'the firewall server version
Global Const FW_VERSION5 = 2
Global Const FW_VERSION4 = 4

' error codes
Global Const ERR_CHANGE_PORT = 1
Global Const ERR_CANNOT_CONNECT = 2
Global Const ERR_CONNECT_TO_RECV = 3
Global Const ERR_CONNECT_TO_POLL = 4
Global Const ERR_CONNECT_TO_SEND = 5
Global Const ERR_NO_HOST_NAME = 6
Global Const ERR_NO_TERM_TYPE = 7
Global Const ERR_UNABLE_TO_LOAD = 8
Global Const ERR_PORT_UNDEFINED = 9
Global Const ERR_FW_SERVER_NOT_DEFINED = 10
```

2. Properties for Telnet

2.1 Action

Summary

Connect, disconnect or abort a Telnet session.

Description

The Action property controls the connection state of the Telnet ActiveX control. A session can be established, closed or aborted by assigning one of the following values to the property.

Value	Meaning
ACTION_CONNECT	Establish session.
ACTION_DISCONNECT	Close session.
ACTION_ABORT	Abort session.
ACTION_FW_CONNECT	Establish session via a firewall.

This property can be changed at run time only.

Before setting the Action property to ACTION_CONNECT, the following properties must be initialized. The HostName property must be set to the name or internet address (in dotted decimal notation) of the Telnet server. The Port property must be set to the remote port on which the Telnet service is running (most servers use the default Telnet service port of 23). The TerminalType property must be set to the type of terminal that will be emulated by the application. The default value "tty" will cause the Telnet server to send no screen control commands. If the application will, for example, emulate a DEC VT100 terminal, then the TerminalType property should be set to "vt100".

If the local machine is located on a different subnet than the remote machine and the only form of communication between these two machines is through a firewall gateway, then the built-in firewall support of the Telnet ActiveX control can be used to established a Telnet session. To establish a connection with a remote host through a firewall, set the Action property to ACTION_FW_CONNECT. Before setting the Action property to ACTION_FW_CONNECT, some properties must be initialized. The FirewallServer property must be set to the name or internet address (in dotted decimal notation) of the firewall server. The FirewallPort property must be set to the firewall service port. In addition to the FirewallServer and FirewallPort properties, the HostName, Port, and TerminalType properties must also be set as mentioned above. Depending on the type of address specified in the HostName property the FwAddrType property must be accordingly set, if the HostName property contains the IP address of the remote host then the FwAddrType must be set to FW_ADDR_IP4 and if contains a machine name then the FwAddrType property must contain FW_ADDR_DNS. The Distinct Telnet ActiveX Control supports both SOCKS version 5 and SOCKS version 4, the application can specify the SOCKS version in the FwSocksVer property. If the SOCKS version is 5 then the application can specify a authentication method in the FwAuthMethods property, currently only the Username/Password (FwUsername and FwPassword) authentication protocol is supported.

If the connection can be established, then the OnConnect event will be fired. If the connection cannot be established, then the OnError event will be fired. These events will occur before the next statement (i.e. the statement following the assignment of ACTION_CONNECT or ACTION_FW_CONNECT to the Action property) is executed. The application should set a flag in the OnConnect and OnError events, so that it can determine if the session has been established or not. In addition, the application may want to display an error message in the OnError event to

inform the user that the connection has not been established. Please check the reference page of the OnError event for a complete listing of error codes.

Once a connection is no longer needed, the session can be terminated by setting the Action property to ACTION_DISCONNECT. An application must close all connections it has created before it quits. A connection can also be closed by setting the Action property to ACTION_ABORT. This action resets and closes the connection without properly closing down and should not be called under normal circumstances.

The Connect, FwConnect, Disconnect and Abort methods accomplish the same as the above actions. Please check the reference pages of these methods for more detailed information on their usage.

There is no default value for this property.

Example

```
Telnet.HostName = "speedy"
Telnet.TerminalType = "vt100"
Telnet.Action = ACTION_CONNECT
```

2.2 BinaryData

Summary

Contains the binary data received following a call to the ReceiveB method.

Description

After each call to the ReceiveB method in a C# application this property needs to be read to access binary data.

Example

```
Private void Telnet_OnReceive (object sender, System.EventArgs e)
{
    int len;
    int bytes;
    Object arrData = new byte[bytes];

    Len = ax Telnet1.ReceiveB (arrData);
    If (len > 0)
    {
        object pBuf = new Byte[len];
        pBuf = ax Telnet1.BinaryData;
        byte []RcvByte = (byte [])pBuf;
        .....
        .....
    }
}
```

2.3 Echo

Summary

Set remote or local echo.

Description

The Echo property controls how local input (any data assigned to the Send property) is echoed back. The property can be set to either one of the following two values.

Value	Meaning
ECHO_REMOTE	Characters are echoed back by server.
ECHO_LOCAL	Application is responsible for displaying characters.

This property can be changed at design time and at run time before a connection has been established.

Most Telnet sessions rely on remote echo. This means that any character sent to the server is echoed back to the application. The application can then display the echoed back message to the user. Remote echo allows the server to translate one character to a different sequence of characters. For example, many Telnet servers translate the ^D character (Control - D) to the command "logout". Also, a control character is often interpreted as a cursor control character. For example, entering ^F (Control - F) during a vi session instructs vi to scroll forward by one page. Obviously, it would not make sense to display the ^F character locally. Instead, it is sent to the server and the server then updates the display with new data without ever echoing back the actual control character.

In some cases, it may be necessary to use local echo. The Telnet server can be instructed that the application will be using local echo by setting the Echo property to ECHO_LOCAL. This will cause the server to not echo back any characters it receives. It may, however, still react to certain cursor control characters. The Echo property can be changed as needed except during a Telnet session.

The default value of this property is ECHO_REMOTE.

Example

Telnet.**Echo** = ECHO_REMOTE

2.4 FirewallPort

Summary

Firewall server port.

Description

The FirewallPort property specifies the port on the firewall server through which a connection can be established. The FirewallPort property must be set before a connection is established (by setting the Action property to ACTION_FW_CONNECT). The Telnet ActiveX control does not verify the setting of the FirewallPort property and any value (including 0) is therefore legal.

This property can be changed at any time except after a connection has been established with a Telnet server (by setting the Action property to ACTION_CONNECT or ACTION_FW_CONNECT). The default value for this property is 1080.

Example

```
Telnet.FirewallServer = "127.43.101.10"  
Telnet.FirewallPort = 1080  
Telnet.HostName = "127.43.101.12"  
Telnet.Port = 23  
Telnet.Action = ACTION_FW_CONNECT
```

2.5 FirewallServer

Summary

Name of firewall server or dotted decimal internet address.

Description

The FirewallServer property specifies the name or internet address of a firewall through which a connection is to be established. This property must be set before a connection can be established (by changing the Action property). There are three possible ways of specifying a firewall server name.

Machine Name

An application only needs to specify the name of the firewall server if the host is located on the same network as the local PC or if the internet address of the host is defined in the local hosts data base. If the firewall server is not on the local network, then the underlying protocol will route the traffic through a gateway. If the host is not defined in the local hosts data base, then the underlying protocol will contact the name server to resolve the internet address of the firewall server.

Machine and Domain Name

An application needs to specify the machine name and the domain name if the host is not located on the same network as the local PC. Fully domain qualified machine names are written from left to right in ascending order (for example, *speedy.distinct.com*). If both machine and domain names are specified, then the underlying protocol will contact the name server to resolve the internet address of the firewall server.

Internet Address

Sometimes the user knows only the internet address of the firewall server that he or she wants to use. In this case, the internet address can be entered in what is known as the dotted decimal notation (for example, *127.43.101.12*). If the host identified by this address is not on the local network, then the underlying protocol will route the traffic through a gateway.

This property can be changed at design time and at run time before a connection has been established. There is no default value for this property.

Example

```
Telnet.FirewallServer = "127.43.101.10"  
Telnet.FirewallPort = 1080  
Telnet.HostName = "127.43.101.12"  
Telnet.Port = 23  
Telnet.Action = ACTION_FW_CONNECT
```

2.6 FwAddrType

Summary

Address type of the destination host.

Description

The *FwAddrType* property specifies the address type in the *Host* property. This property must be set before a connection can be established (by changing the Action property to ACTION_FW_CONNECT or calling the method FwConnect). There are three possible ways of specifying a destination address in the *Host* Property and therefore the *FwAddrType* property can have three possible values:

FW_ADDR_IP4	Address is a version 4 IP address
FW_ADDR_DNS	Address is a DNS style domain name
FW_ADDR_IP6	Address is a version 6 IP address

The *FwAddrType* property must be set to FW_ADDR_IP4 if the application is specifying a Internet address in the dotted decimal notation (*198.211.122.133*) in the *Host* property. If the application wants to specify the machine name or machine name and domain name (for example *speedy.distinct.com*) as the *Host* they must set the *FwAddrType* property to FW_ADDR_DNS.

Note that the FW_ADDR_IP6 is not supported currently.

This property can be changed at design time and at run time before a connection has been established.

The default value for this property is FW_ADDR_IP4.

Example

```
Telnet.FirewallServer = "sparky.distinct.com"
Telnet.FirewallPort = 1080
Telnet.FwAddrType = FW_ADDR_DNS
Telnet.HostName = "speedy.distinct.com"
Telnet.Action = ACTION_FW_CONNECT
```

2.7 FwAuthMethods

Summary

Firewall authentication methods.

Description

The *FwAuthMethods* property specifies the authentication method that can be used when connecting to the firewall server. This property must be set before a connection can be established (by changing the Action property to ACTION_FW_CONNECT or calling the method FwConnect). The *FwAuthMethods* can be "0", "1" or "2" or a combination of "0", "1", "2", for example "01" or "12". "0" means that no authentication is required, "1" means GSSAPI and "2" means that a valid username and password is required. Currently only methods "0" and "2" are supported.

This property can be changed at design time and at run time before a connection has been established.

The default value for this property is "0".

Example

```
Telnet.FirewallServer = "sparky.distinct.com"  
Telnet.FirewallPort = 1080  
Telnet.FwAddrType = FW_ADDR_DNS  
Telnet.HostName = "speedy.distinct.com"  
Telnet.FwAuthMethods = "2"  
Telnet.FwUsername = "joe"  
Telnet.FwPassword = "distinct"  
Telnet.Action = ACTION_FW_CONNECT
```

2.8 FwPassword

Summary

A valid password .

Description

The *FwPassword* property specifies a valid password that is required during authentication. A valid password is required when connecting to SOCKS version 5 server if the authentication method (*AuthMethod*) specified is "2" (Username/Password authentication protocol).

This property can be changed at design time and at run time before a connection has been established by setting the *Action* property to ACTION_FW_CONNECT or by calling the method *FwConnect*.

The property does not have any default value.

Example

```
Telnet.FirewallServer = "sparky.distinct.com"  
Telnet.FirewallPort = 1080  
Telnet.FwAddrType = FW_ADDR_DNS  
Telnet.HostName = "speedy.distinct.com"  
Telnet.FwAuthMethods = "2"  
Telnet.FwUsername = "joe"  
Telnet.FwPassword = "distinct"  
Telnet.Action = ACTION_FW_CONNECT
```

2.9 FwSocksVer

Summary

The firewall server version.

Description

The *FwSocksVer* property is used specify the version of the SOCKS server. This property can have any one or a combination of the following values.

Value	Meaning
FW_VERSION5	The SOCKS version is 5.
FW_VERSION4	The SOCKS version is 4

If the firewall server version is unknown then the application can specify both FW_VERSION5 and FW_VERSION4. The Distinct Telnet control will automatically detect the firewall server version and make the appropriate connection.

This property can be set at design time or at run time before a connection is established by calling the method FwConnect or setting the Action to ACTION_FW_CONNECT.

Example

```
Telnet.FirewallServer = "sparky.distinct.com"  
Telnet.FirewallPort = 1080  
Telnet.FwAddrType = FW_ADDR_DNS  
Telnet.HostName = "speedy.distinct.com"  
Telnet.FwAuthMethods = "2"  
Telnet.FwUsername = "joe"  
Telnet.FwPassword = "distinct"  
Telnet.FwSocksVer = FW_VERSION5  
Telnet.Action = ACTION_FW_CONNECT
```

2.10 FwUsername

Summary

A valid user name.

Description

The *Username* property specifies a valid username. A valid user id is required when connecting to SOCKS version 5 server if the authentication method (*AuthMethods*) specified is "2" (Username/Password authentication protocol). It is mandatory when a connection needs to established with SOCKS version 4 server.

This property can be changed at design time and at run time before a connection has been established by setting the Action property to ACTION_FW_CONNECT or by calling the method FwConnect.

The property does not have any default value.

Example

```
Telnet.FirewallServer = "sparky.distinct.com"  
Telnet.FirewallPort = 1080  
Telnet.FwAddrType = FW_ADDR_DNS  
Telnet.HostName = "speedy.distinct.com"  
Telnet.FwAuthMethods = "2"  
Telnet.FwUsername = "joe"  
Telnet.FwPassword = "distinct"  
Telnet.Action = ACTION_FW_CONNECT
```

2.11 HostName

Summary

Name of host or dotted decimal internet address.

Description

The HostName property specifies the name or internet address of a Telnet server. This property must be set before a session can be established. There are three possible ways of specifying a Telnet server.

Machine Name

An application only needs to specify the name of the Telnet server if the server is located on the same network as the local PC or if the internet address of the server is defined in the local host table. If the Telnet server is not on the local network, then the underlying protocol will route the traffic through a gateway. If the Telnet server is not defined in the local host table, then the underlying protocol will contact the domain server to resolve the internet address of the server.

Machine and Domain Name

An application needs to specify the machine name and the domain name if the Telnet server is not located on the same network as the local PC. Fully domain qualified machine names are written from left to right in ascending order (for example, *speedy.distinct.com*). If both machine and domain names are specified, then the underlying protocol will contact the domain server to resolve the internet address of the server.

Internet Address

Sometimes the user knows only the internet address of the Telnet server that he or she wants to use. In this case, the internet address can be entered in what is known as the dotted decimal notation (for example, *127.43.101.12*). If the Telnet server identified by this address is not on the local network, then the underlying protocol will route the traffic through a gateway.

This property can be changed at design time and at run time before a connection has been established. There is no default value for this property.

Example

```
Telnet.HostName = "127.43.101.12"  
Telnet.TerminalType = "vt100"  
Telnet.Action = ACTION_CONNECT
```

2.12 Port

Summary

Telnet service port on host.

Description

The Port property specifies the remote port on the Telnet server on which the Telnet service resides. Most Telnet services listen for connection requests on port 23. Sometimes, usually for security reasons, port numbers other than 23 are used for Telnet connections.

If the application will be connecting to Telnet services on a different port, then the Port property must be set before the connection attempt is made. An application may even want to query the user for the correct port before connecting. The ActiveX control does not verify the setting of the Port property and any value is therefore legal.

This property can be changed at design time and at run time before a connection has been established. The default value for this property is port 23.

Example

```
Telnet.HostName = "127.43.101.12"  
Telnet.TerminalType = "vt100"  
Telnet.Port = 23  
Telnet.Action = ACTION_CONNECT
```

2.13 Receive

Summary

Receive buffer.

Description

The Receive property is used to access data that has been sent by the Telnet server. This property can only be accessed by assigning its value to another string while a session is established. Whenever the property is assigned to a string, as many bytes of data as possible up to the number specified by the ReceiveLen property are copied from the receive buffer into that string.

Usually, this property is read during an OnReceive event. This event informs the application that more data has arrived from the server. At this point, the application should read all the data available and process it. The ReceiveCount property can be used to find out how many total bytes of data are waiting in the receive buffer.

If the ReceiveLen property is set to a value less than the ReceiveCount property, then the application may have to read the Receive property more than once to read all available data. In general, the application should get data from the receive buffer until no more data is available. This condition can be detected by checking the length of the string to which the Receive property is assigned. As soon as this length is zero, all data has been read.

Since the ReceiveLen property can be changed at any time (even while connected), an application could assign the value of the ReceiveCount property to ReceiveLen. Any subsequent assignment of the Receive property would then copy all available data from the receive buffer into the application's buffer. This approach is not advisable if very large amounts of data are being received.

The Result property will contain the total number of bytes read from the receive buffer. The Result property will contain a value less than or equal to the value specified by the ReceiveLen property.

This property can only be read at run time while a connection is established. There is no default value for this property.

Example

```
Dim Message As String
```

```
Message = Telnet.Receive
```

2.14 ReceiveCount

Summary

Number of bytes waiting in receive buffer.

Description

The ReceiveCount property specifies how many bytes of data are available in the receive buffer. This property is read only and can only be accessed while a session is established.

Because of the interrupt driven nature of network communications, the value of the ReceiveCount property can change quickly at any time. An application should not rely on this value for an extended period of time without rechecking it.

The receive buffer can be accessed with the Receive property. The number of bytes read from the receive buffer while accessing the Receive property can be set with the ReceiveLen property.

This property can only be read at run time while a connection is established. There is no default value for this property.

Example

```
If Telnet.ReceiveCount > 0 Then
    MsgBox "Data is now available in the Receive buffer", 64, "Sample Program"
End If
```

2.15 ReceiveLen

Summary

Maximum number of bytes to read from receive buffer.

Description

The ReceiveLen property determines the maximum number of bytes of data that are read from the receive buffer when an application accesses the Receive property.

The default value of 100 should be suitable for most applications, but any value other than zero is legal. Setting the ReceiveLen property value to one will cause the application to read one byte of data at a time. If an application wants to make sure that it reads the total number of bytes available, then it should set the value of the ReceiveLen property equal to the value of the ReceiveCount property before accessing the Receive property.

The receive buffer can be accessed with the Receive property. The total number of bytes waiting in the receive buffer can be determined by checking the ReceiveCount property.

This property can be changed at any time. The default value for this property is 100 bytes.

Example

Telnet.**ReceiveLen** = 25

2.16 Result

Summary

Number of bytes sent or received.

Description

The Result property contains the number of bytes that were sent during a send operation and the number of bytes read during a receive operation. This property can only be read by the application and cannot be changed.

Data is sent by assigning a string to the Send property. Care should be taken not to exceed the buffer and transport capabilities of the underlying protocol stack. When using the TCP protocol, in general, no more than 512 bytes of data should be assigned to the Send property at any one time.

The Result property must be checked each time after assigning data to the Send property or after accessing the Receive property. If an error occurs during the send operation, then the Result property will contain the number of bytes that were successfully sent before the error occurred. Hence, this property may be used to verify whether all of the data or only a portion has been sent to the remote host. The Error property will contain any error code returned by the protocol stack.

Data is received by accessing the Receive property. This property can only be accessed by assigning its value to another string while a connection is established. Whenever the property is assigned to a string, as many bytes of data as possible up to the number specified by the ReceiveLen property are copied from the receive buffer into that string.

The Result property must be checked each time the Receive property is accessed to read the data sent by the remote machine. The ReceiveLen property specifies the maximum number of bytes that can be read from the receive buffer and the Result property specifies the exact number of bytes that were read from the receive buffer. The Result property will always contain a value less than or equal to the value specified by the ReceiveLen property.

This property can only be read at run time while a connection is established. There is no default value for this property.

Example

```
Telnet.Send = "This is a test"  
If Telnet.Result = 14 Then  
    MsgBox "Message sent successfully", 64, "Sample Program"  
End If
```

2.17 Send

Summary

Send buffer.

Description

The Send property is used to transfer data to the Telnet server. Data is sent by assigning a string to this property. Care should be taken not to exceed the buffer and transport capabilities of the underlying protocol stack. In general, no more than 512 bytes of data should be assigned to this property at any one time. Most TCP/IP stacks are able to transfer multiple 512 byte chunks of data in quick succession.

Usually, this property is set in response to user input (for example, entering a command to be executed on the Telnet server). The response to these user commands can be read from the Receive property in response to OnReceive events. The ReceiveCount property can be used to check how much data (if any) is available to be read.

The Result property will contain the total number of bytes sent. If no error occurred, then the Result property will be equal to the number of bytes assigned to the Send property.

This property can only be written to at run time while a connection is established. There is no default value for this property.

Example

```
Telnet.Send = "This is a test"
```

2.18 TerminalType

Summary

Type of terminal emulated by the application.

Description

The TerminalType property specifies the type of terminal emulation the application will perform. This property must be set to a string containing at least one character before a connection is established. Applications not performing any terminal emulation should keep the default value of this property.

The Telnet ActiveX control does not perform any type of terminal emulation. Its purpose is to shield the calling application from the Telnet protocol and various options involved in establishing a connection. If an application will perform terminal emulation, then it must set the TerminalType property to the type of terminal it will emulate. For example, an application emulating a DEC VT100 terminal would set this property to "vt100".

This property can be changed at design time and at run time before a connection is established. The default value for this property is "tty".

Example

```
Telnet.HostName = "speedy.distinct.com"  
Telnet.TerminalType = "vt100"  
Telnet.Action = ACTION_CONNECT
```

3 Events for Telnet

3.1 OnClose

Summary

Telnet connection has been closed.

Description

The OnClose event occurs usually in response to setting the Action property to ACTION_DISCONNECT or ACTION_ABORT (or in response to the Disconnect or Abort method). In most cases, the remote server may close a connection. This will also trigger an OnClose event. In this case, the application must still set the Action property to ACTION_DISCONNECT (or use the Disconnect method) to free up all the resources allocated for the connection. However, this should be done with care during the OnClose event because it might result in an infinite loop.

If this event occurs in response to setting the Action property to ACTION_DISCONNECT (or in response to the Disconnect method), it will occur before the statement following the assignment of ACTION_DISCONNECT to the Action property (or the call to the Disconnect method) is reached.

Normally, an application should simply set a flag in response to this event. Then, this flag can be checked directly after the ACTION_DISCONNECT action (or the Disconnect method) to make sure that the connection was actually terminated.

Example

```
Sub Telnet_OnClose ()
    If Connected = True Then
        Connected = False
        Telnet.Action = ACTION_DISCONNECT
    End If
End Sub
```

3.2 OnConnect

Summary

Telnet connection has been established.

Description

The OnConnect event occurs in response to setting the Action property to ACTION_CONNECT (or in response to the Connect method). This event will occur before the statement following the assignment of ACTION_CONNECT to the Action property (or the call to the Connect method) is reached.

Normally, an application should simply set a flag in response to this event. Then, this flag can be checked directly after the ACTION_CONNECT action (or the Connect method) to make sure that the connection was actually established.

If the connection could not be established, then the OnError event will be called instead of the OnConnect event.

While handling the OnConnect event, an application should not perform tasks which have the potential of requiring a lot of time to complete, such as generating a message box.

Example

```
Sub Telnet_OnConnect ()  
    Connected = True  
End Sub
```

3.3 OnError

Summary

Local error has occurred.

Description

The OnError event occurs when a property is accessed in an illegal way or when a connection with the Telnet server cannot be established. The table below lists all possible error codes delivered by this event.

Value	Meaning
ERR_CHANGE_PORT	Cannot change port while connected.
ERR_CANNOT_CONNECT	Unable to connect to remote host.
ERR_CONNECT_TO_RECV	Must be connected before accessing receive buffer.
ERR_CONNECT_TO_POLL	Must be connected before accessing receive count.
ERR_CONNECT_TO_SEND	Must be connected before accessing send buffer.
ERR_NO_HOST_NAME	Host name must be defined before connecting.
ERR_NO_TERM_TYPE	Terminal type must be defined before connecting.

The following describes each error in more detail.

ERR_CHANGE_PORT

An attempt was made to change the Port property while connected.

ERR_CANNOT_CONNECT

Telnet server is unreachable. The Port or the HostName properties may be set incorrectly or the host may be down.

ERR_CONNECT_TO_RECV

An attempt was made to access the Receive property without being connected.

ERR_CONNECT_TO_POLL

An attempt was made to access the ReceiveCount property without being connected.

ERR_CONNECT_TO_SEND

An attempt was made to access the Send property without being connected.

ERR_NO_HOST_NAME

The HostName property was not set before attempting to establish a connection.

ERR_NO_TERM_TYPE

The TerminalType property was not set before attempting to establish a connection.

Example

```
Sub Telnet_OnError (ErrorCode As Integer)
  If ErrorCode = ERR_CANNOT_CONNECT Then
    MsgBox "Unable to connect to remote host", 64, "Sample Program"
  End If
End Sub
```

3.4 OnReceive

Summary

More data has been received.

Description

The OnReceive event occurs whenever the underlying stack receives one or more bytes of additional data from the Telnet server. The event does not actually deliver the data to the application but instead expects that the application will access the Receive property to obtain all data waiting in the receive buffer.

The ReceiveCount property can be accessed to determine how many bytes of data are actually available. The ReceiveLen property specifies how many bytes of data will be obtained from the receive buffer every time the Receive property is accessed.

The Result property will contain the total number of bytes read from the receive buffer. The Result property will contain a value less than or equal to the value specified by the ReceiveLen property.

An application does not have to rely on the OnReceive event to notify it of incoming data. Instead of or in addition to this event, the application may check the value of the ReceiveCount property to find out if any data is available.

For more information on how to receive data, please check the reference pages of the Receive, ReceiveCount and ReceiveLen properties.

While handling the OnReceive event, an application should not perform tasks which have the potential of requiring a lot of time to complete, such as generating a message box. A substantial delay could cause the application to not receive subsequent OnReceive events.

Example

```
Sub Telnet_OnReceive ()  
    Dim Message As String  
  
    Message = Telnet.Receive  
End Sub
```

4. Methods for Telnet

4.1 Abort

Summary

Abort a remote session.

Syntax

Boolean Abort ()

Description

The Abort method aborts a session to the remote server. This method resets and closes the connection without properly closing down and should not be called under normal circumstances.

The Abort method takes no parameters and returns a boolean. If the connection is successfully reset and closed, then the method returns True; otherwise, it returns False. The application should ensure that the method was successfully executed by checking the return value.

Calling this method is equivalent to setting the Action property to ACTION_ABORT.

Example

```
Result = Telnet.Abort ()  
If Result = False Then  
    MsgBox "Unable to abort", 64, "Sample Program"  
End If
```

4.2 Connect

Summary

Connect to the remote server.

Syntax

Boolean Connect (*Host*, *Tty*)

Host String

Tty String

Description

The Connect method establishes a connection to the remote server.

The Connect method takes a host name (*Host*) and a terminal type (*Tty*) as its parameters and returns a boolean. The host name must be the name or internet address (in dotted decimal notation) of the remote server. The terminal type must be set to the type of terminal that will be emulated by the application. The value "tty" will cause the Telnet server to send no screen control commands. If the application will, for example, emulate a DEC VT100 terminal, then the tty string should be set to "vt100". If a connection is successfully established, then the method returns True; otherwise, it returns False. The application should ensure that the method was successfully executed by checking the return value.

If the connection can be established, then the OnConnect event will be fired. If the connection cannot be established, then the OnError event will be fired. These events will occur before the next statement (i.e. the statement following the call to the Connect method) is executed. The application should set a flag in the OnConnect and OnError events, so that it can determine if the session has been established or not. In addition, the application may want to display an error message in the OnError event to inform the user that the connection has not been established. Please check the reference page of the OnError event for a complete listing of error codes.

Calling this method is equivalent to setting the Action property to ACTION_CONNECT.

Example

```
Result = Telnet.Connect ("speedy.distinct.com", "tty")
If Result = False Then
    MsgBox "Unable to connect to server", 64, "Sample Program"
End If
```

4.3 Disconnect

Summary

Close connection to the remote server.

Syntax

Boolean Disconnect ()

Description

Once a connection is no longer needed, the session can be terminated by calling the Disconnect method. An application must close all connections it has created before it quits.

The Disconnect method takes no parameters and returns a Boolean. If a connection is successfully terminated, then the method returns True; otherwise, it returns False. The application should ensure that the method was successfully executed by checking the return value.

Calling this method is equivalent to setting the Action property to ACTION_DISCONNECT.

Example

```
Result = Telnet.Disconnect ()  
If Result = False Then  
    MsgBox "Unable to disconnect from server", 64, "Sample Program"  
End If
```

4.4 FwConnect

Summary

Establish session via a firewall.

Syntax

Boolean FwConnect (*FwServer*, *FwPort*, *Host*, *Tty*)

<i>FwServer</i>	String
<i>FwPort</i>	Integer
<i>DestHost</i>	String
<i>Tty</i>	String

Description

The FwConnect method establishes a Telnet session via a firewall.

If the local machine is located on a different subnet than the remote Telnet server machine and the only form of communication between these two machines is through a firewall gateway, then the built-in firewall support of the Telnet ActiveX control can be used to establish a Telnet session.

The FwConnect method takes a firewall server name (*FwServer*), a firewall port (*FwPort*), a Telnet host (*Host*), and the type of terminal emulation (*Tty*) as its parameters and returns a boolean. The Port property must be set to the remote port on the Telnet server on which the Telnet service resides. The firewall server string must be set to the name or internet address of the firewall through which connection is to be established. The firewall port integer must be set to the port on which the firewall server is listening for connections. The host string must be set to the name or internet address of the Telnet server. If the *Host* is the internet address then the *FwAddrType* property must be set to FW_ADDR_IP4 and if it is a machine name then the *FwAddrType* property must be set to FW_ADDR_DNS, by default the *FwAddrType* property has the value FW_ADDR_IP4. The terminal type must be set to the type of terminal emulation the application will perform.

The application can also set the *FwAuthMethods* property if it wants to specify any authentication that needs to be negotiated before an actual connection is established. Only the Username/Password authentication is currently supported. If Username/Password authentication is specified the *FwUsername* and *FwPassword* properties should contain a valid username and password respectively. An authentication is only possible if the SOCKS version is 5; version 4 SOCKS server does not support any authentication protocols. If the SOCKS version is 4 then the application must provide a valid user id (*FwUsername*).

The version of the SOCKS server can be specified in the *FwSocksVer* property. If the version is unknown then the application can set the *FwSocksVer* Property to FW_VERSION4 and FW_VERSION5 and the Distinct Telnet ActiveX control will automatically detect the SOCKS version and connect appropriately.

If a connection is successfully established, then the method returns True; otherwise, it returns False. The application should ensure that the method was successfully executed by checking the return value.

If the connection can be established, then the OnConnect event will be fired. If the connection cannot be established, then the OnError event will be fired. These events will occur before the next statement (i.e. the statement following the call to the FwConnect method) is executed. The application should set a flag in the OnConnect and OnError events, so that it can determine if the session has been established or not. In addition, the application may want to display an error message in the OnError event to inform the user that the connection has not been established. Please check the reference page of the OnError event for a complete listing of error codes.

Calling this method is equivalent to setting the Action property to ACTION_FW_CONNECT.

Example

```
Result = Telnet.FwConnect ("sparky.distinct.com", 1080, "speedy.distinct.com", "tty")
If Result = False Then
    MsgBox "Unable to connect", 64, "Sample Program"
End If
```

4.5 ReceiveB

Summary

Receive binary data.

Syntax

Long ReceiveB (*Buffer*)
Buffer Variant

Description

The ReceiveB method is used to access binary data that has been sent by the Telnet server. Whenever this method is called, as many bytes of data as possible (up to the number specified by the ReceiveLen property) are copied to the buffer parameter passed to it.

The ReceiveB method takes a variant buffer (*Buffer*) as its parameter and returns a long. If operation is successful, then the method fills the variant buffer with data received and returns the number of bytes that was received; otherwise, it sets the buffer to NULL and returns 0. The application should ensure that the method was successfully executed by checking the variant buffer or the return value.

Usually, this method is called during an OnReceive event. This event informs the application that more data has arrived from the connected host. At this point, the application should read all the data available and process it. The ReceiveCount property can be used to find out how many total bytes of data are waiting to be read.

If the ReceiveLen property is set to a value less than the ReceiveCount property, then the application may have to call the ReceiveB method more than once to read all available data. In general, the application should get data from the receive buffer until no more data is available. This condition can be detected by checking the value returned by the ReceiveB method. As soon as this value is zero, all data has been read.

Since the ReceiveLen property can be changed at any time (even while connected), an application could assign the value of the ReceiveCount property to ReceiveLen. Any subsequent call to the ReceiveB method would then copy all available data to the buffer parameter passed to the method. This approach is not advisable if very large amounts of data are being received.

The Result property and the return value of the ReceiveB method will contain the total number of bytes read. The Result property will contain a value less than or equal to the value specified by the ReceiveLen property.

This method can only be read at run time while a connection is established.

Example

```
Dim Buffer() As Byte
Dim Siz As Long
Dim i As Integer

Siz = Telnet.ReceiveB (Buffer)
If Siz > 0 Then
    Open "c:\abc.exe" For Binary Access Write As #1
    For i = 1 To Siz
        Put #1, i, Buffer (i)
    Next i
    Close #1
Else
```

```
    MsgBox "Cannot receive binary data", 64, "Sample Program"  
End If
```

4.6 SendB

Summary

Send binary data.

Syntax

Boolean SendB (*Buffer*, *Bytes*)

Buffer Variant

Bytes Long

Description

The SendB method is used to transfer binary data to the Telnet server over an established connection. Care should be taken not to exceed the buffer and transport capabilities of the underlying protocol stack. In general, no more than 512 bytes of data should be assigned to this property at any one time. Most TCP/IP stacks are able to transfer multiple 512 byte chunks of data in quick succession.

The SendB method takes a variant buffer (*Buffer*) and a long byte (*Bytes*) as its parameters and returns a boolean. The variant buffer has to be set to the data to be sent. The bytes parameter has to be set to the number of bytes of data (or the size of the buffer parameter) to be sent. If the operation is successful, then the method returns True; otherwise, it returns False. The application should ensure that the method was successfully executed by checking the return value.

The Result property will contain the total number of bytes sent. If no error occurred, then the Result property will be equal to the number of bytes passed to the SendB method.

Incoming data can be read by calling the ReceiveB method in response to OnReceive events. The ReceiveCount property can be used to check how much data (if any) is available to be read.

This method can only be call at run time while a connection is established.

Example

```
Dim Buffer() As Byte
Dim Bytes As Long
Dim i As Integer

Open "c:\abc.exe" For Binary Access Read As #1
Bytes = FileLen("c:\abc.exe")
For i = 1 To Bytes
    Get #1, , Buffer(i)
Next i
Result = Telnet.SendB(Buffer, Bytes)
If Result = False Then
    MsgBox "Cannot send binary data", 64, "Sample Program"
End If
Close #1
```

Registry Entries

Telnet Client Registry Entries

You can change the values of the connection timeout and buffer size used for option negotiations for Distinct Telnet library by making registry entries under the following registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Distinct\DLLS\TNET32

Timeout REG_DWORD Number

Specifies the default timeout in seconds used for the Telnet connection. This is the timeout value used for various network operations. The default value is 20 seconds.

BufferSize REG_DWORD 128-8192 bytes

Specifies the default buffer size in bytes used for option negotiations. The default value is 1024 bytes.