

# **Firewall ActiveX Control**

**SOCKS**

**ActiveX Control**

**for Microsoft® Windows™**

**Version 5.2**

**Copyright © 1996 - 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](mailto: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**

© 1996 - 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 Firewall 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.....	3
<b>1 Overview</b> .....	5
1.1 Introduction.....	5
1.2 Usage.....	5
1.3 Property Summary.....	6
1.4 Event Summary.....	7
1.5 Method Summary.....	8
1.6 D_FW.TXT.....	8
<b>2 Properties</b> .....	11
2.1 Action.....	11
2.2 AddrType.....	13
2.3 AuthMethods.....	14
2.4 BinaryData.....	15
2.5 Connection.....	16
2.6 DestHost.....	17
2.7 DestPort.....	18
2.8 FirewallPort.....	19
2.9 FirewallServer.....	20
2.10 LastResult.....	21
2.11 Password.....	23
2.12 Send.....	24
2.13 SocksVer.....	25
2.14 Status.....	26
2.15 Username.....	27
2.16 UseVariant.....	28
<b>3 Events</b> .....	29
3.1 OnClose.....	29
3.2 OnConnect.....	30
3.3 OnError.....	31
3.4 OnFwListen.....	33
3.5 OnReceive.....	34
3.6 OnReceiveB.....	35
3.7 OnRemote.....	36
<b>4 Methods</b> .....	37
4.1 Abort.....	37
4.2 AsyncConnect.....	38
4.3 Close.....	40
4.4 Connect.....	41
4.5 Listen.....	42
4.6 ReceiveB.....	43
4.7 Select.....	44
4.8 SendB.....	45



## 1 Overview

### 1.1 Introduction

The Distinct Firewall ActiveX control allows you to write applications that securely traverse a network firewall. This is vital, as most companies now have a security firewall that controls the flow of packets in and out of the local network. Without firewall support your application would be unable to communicate with any computer outside the local network. This ActiveX control supports SOCKS 5 and SOCKS 4. SOCKS is a networking proxy protocol that enables hosts on one side of the SOCKS server to access hosts on the other side of the SOCKS server without direct IP reachability. SOCKS support is also built into the FTP Client, NNTP, POP, SMTP, Telnet and Windows Sockets ActiveX controls.

### 1.2 Usage

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

After placing a Firewall 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 FirewallPort property defaults to 1080, which is the most commonly used value. The DestPort property determines the port on the remote machine with which a connection should be established. The FirewallServer property specifies the internet address or host name of the firewall server. Please check the reference pages of these properties for more details.

The DestHost 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 remote server from the user. If an application will always connect to the same host, then the DestHost property can also be set at design time to minimize user interaction.

The Firewall ActiveX control supports the Username/Password authentication protocol. If the application chooses to authenticate the user before actually establishing a connection with the server it must set the AuthMethods property to "2" , the Username and Password properties to a valid user id and password respectively.

Since the Firewall ActiveX control supports both Socks version 5 and version 4 the application is given an option to specify the Socks version in the SocksVer property. If the application is not sure about the server version then they can leave the property empty and the Firewall ActiveX control will automatically determine the server version.

The Action property can only be accessed at run time. To establish a session with the remote server via a firewall, the value of the Action property must be set to ACTION\_CONNECT (or the Connect method must be called) or ACTION\_ASYNC\_CONNECT (or the AsyncConnect method must be called). The FirewallServer, FirewallPort, DestHost and DestPort properties must be initialized before connecting. If the connection can be established, the OnConnect event will occur before the next line of code is reached. The Connection property can be read to get the handle of the connected socket.

Once a connection has been established, data can be sent with the Send property and available data can be received through one or more OnReceive events.

The status of a connection can be obtained by setting the Action property to ACTION\_SELECT (or by calling the Select method). This operation sets the status of the connection to the Status property.

A secondary connection can be established after a primary connection (by setting the Action property to ACTION\_CONNECT or by calling the Connect method) has been established with the remote server. To open a secondary connection, set the Action property to ACTION\_LISTEN (or call the Listen method). If the action is successful, the OnFwListen event is fired.

A connection can be aborted by setting the action property to ACTION\_ABORT (or the Abort method can be called).

If a connection is no longer needed, then the application must close the connection by setting the Action property to ACTION\_CLOSE (or by calling the Close method). Once the connection is closed, the OnClose event will occur before the next line of code is reached. Applications must close the connection before terminating.

### 1.3 Property Summary

#### Action

Connect to a remote server via a firewall, close a connection to the firewall, open a secondary connection, or check connection status

#### AddrType

The address type of the destination host

#### AuthMethods

Authentication methods

#### BinaryData

Contains the binary data received following a call to ReceiveB (C#)

**Connection**

Current connection handle

**DestHost**

Remote host name

**DestPort**

Remote port to be used for next connection

**FirewallPort**

Port firewall server listening on

**FirewallServer**

Firewall server name

**LastResult**

Result of the last action

**Password**

A valid password

**Send**

Send buffer

**SocksVer**

The socks server version

**Status**

Connection status

**Username**

A valid user id

**UseVariant**

Send and receive binary or ascii data

## 1.4 Event Summary

**OnClose**

Connection has been closed

**OnConnect**

Connection has been established

**OnError**

Local error has occurred

**OnFwListen**

Secondary connection has been established

**OnReceive**

Data has arrived

**OnReceiveB**

Data has arrived and it is to be read as binary

**OnRemote**

Connection request has been received

## 1.5 Method Summary

### Abort

Abort the current action

### AsyncConnect

Asynchronously connect to the remote host via the firewall

### Close

Close connection to remote host via firewall

### Connect

Connect to remote host via firewall

### Listen

Open a secondary connection

### ReceiveB

Reads binary data

### Select

Retrieve status of connection

### SendB

Send binary data

## 1.6 D\_FW.TXT

The following provides a complete listing of the D\_FW.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).

```
' Firewall ActiveX Control
' (C) Copyright 1996 - 1998 by Distinct Corporation
' All rights reserved

' actions
Global Const ACTION_NONE = 0
Global Const ACTION_CONNECT = 1
Global Const ACTION_LISTEN = 2
Global Const ACTION_CLOSE = 3
Global Const ACTION_SELECT = 4
Global Const ACTION_ASYNC_CONNECT = 5
Global Const ACTION_ABORT = 6

'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_FW_SERVER_NOT_DEFINED = 1
Global Const ERR_DEST_HOST_NOT_DEFINED = 2
Global Const ERR_IN_ACTION = 3
Global Const ERR_CANNOT_CONNECT = 4
Global Const ERR_CANNOT_LISTEN = 5
Global Const ERR_CANNOT_CLOSE = 6
Global Const ERR_CANNOT_CHANGE_FW_PORT = 7
Global Const ERR_CANNOT_CHANGE_DEST_PORT = 8
Global Const ERR_FW_PORT_NOT_DEFINED = 9
Global Const ERR_DEST_PORT_NOT_DEFINED = 10
Global Const ERR_NOT_CONNECTED = 11
Global Const ERR_CANNOT_SEND = 12
Global Const ERR_BIND_FAILED = 13
Global Const ERR_CONNECT_FAILED = 14
```



## 2. Properties

### 2.1 Action

#### Summary

Connect to a remote server via a firewall, close a connection to the firewall, open a secondary connection, or check connection status.

#### Description

The Action property controls the connection state of the Firewall ActiveX Control. The application can connect to a remote server via a firewall, close the established connection, open a secondary connection with the remote server, and obtain the status of a connection by assigning one of the following values to the property.

Value	Meaning
ACTION_ABORT	Abort the current action
ACTION_ASYNC_CONNECT	Asynchronously connect to the remote host via the firewall
ACTION_CONNECT	Connect to the remote host via the firewall server.
ACTION_LISTEN	Establish a secondary connection with the remote host.
ACTION_CLOSE	Close the firewall connection.
ACTION_SELECT	Determine the status of a connection.

This property can be changed at run time only.

The Firewall ActiveX control can establish a connection with a remote host via the firewall server by setting the Action property to ACTION\_CONNECT or ACTION\_ASYNC\_CONNECT. The FirewallServer, FirewallPort, DestHost, and DestPort properties must be set before setting the Action property to ACTION\_CONNECT or ACTION\_ASYNC\_CONNECT. The FirewallServer must contain the host name or internet address of the firewall server. The FirewallPort has to be set to the port the firewall server is listening for connections. The DestHost must contain the host name or internet Address of the remote host. The DestPort must be set to the port of the remote host with which a connection is to be established. The application can also set the AuthMethods property if it wants the firewall ActiveX control to negotiate any of the supported authentications before establishing a connection. If the application chooses the Username/Password authentication protocol then it should set the Username and Password properties. The OnConnect event is generated in response to a successful ACTION\_CONNECT or ACTION\_ASYNC\_CONNECT action.

To check on the status of a connection, set the Action property to ACTION\_SELECT. This action will set the Status property with the result.

When a secondary connection is needed after a primary connection has been made with the Action property set to ACTION\_CONNECT, set the Action property to ACTION\_LISTEN. A secondary connection is used in protocols which require the local machine to accept connections from the remote server. FTP is a well-known example.

Before setting the Action property to ACTION\_LISTEN, the FirewallServer, FirewallPort, DestHost, and DestPort properties must be set. The FirewallServer must contain the IP address or host name of the firewall server. The FirewallPort has to be set to the port the firewall server is listening for connections. The DestHost must contain the IP Address or host name of the local machine. The DestPort must be set to the port the local host is listening for connections. If the

secondary connection is successfully established, the OnFwListen event is fired. When a connection request has been received, the OnRemote event is fired.

If a connection is no longer needed or the application wants to close an established connection, then the Action property must be set to ACTION\_CLOSE. If a secondary connection was opened, that connection will also be closed. Once the connection is closed, the OnClose event will occur before the next line of code is reached. An application must close all connections it has created before it quits.

An action can be aborted by setting the Action property to ACTION\_ABORT or by calling the method Abort.

There is no default value for this property.

### Example

```
Firewall.FirewallServer = "sparky.distinct.com"  
Firewall.FirewallPort = 1080  
Firewall.AddrType = FW_ADDR_DNS  
Firewall.DestHost = "speedy.distinct.com"  
Firewall.DestPort = 13  
Firewall.Action = ACTION_CONNECT
```

## 2.2 AddrType

### Summary

Address type of the destination host.

### Description

The AddrType property specifies the address type in the DestHost. This property must be set before a connection can be established (by changing the Action property). There are three possible ways of specifying a destination address in the DestHost Property and therefore the AddrType 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 AddrType property must be set to FW\_ADDR\_IP4 if the application is specifying an Internet address in the dotted decimal notation (*198.211.122.133*) in the DestHost property. If the application wants to specify the machine name or machine name and domain name (for example *speedy.distinct.com*) as the DestHost they must set the AddrType 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

```
Firewall.FirewallServer = "sparky.distinct.com"  
Firewall.FirewallPort = 1080  
Firewall.AddrType = FW_ADDR_DNS  
Firewall.DestHost = "speedy.distinct.com"  
Firewall.DestPort = 13  
Firewall.Action = ACTION_CONNECT
```

## 2.3 AuthMethods

### Summary

Firewall authentication methods.

### Description

The AuthMethods 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). The AuthMethods 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

```
Firewall.FirewallServer = "sparky.distinct.com"  
Firewall.FirewallPort = 1080  
Firewall.AddrType = FW_ADDR_DNS  
Firewall.DestHost = "speedy.distinct.com"  
Firewall.DestPort = 13  
Firewall.AuthMethods = "2"  
Firewall.Username = "joe"  
Firewall.Password = "distinct"  
Firewall.Action = ACTION_CONNECT
```

## 2.4 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 Fw_OnReceive (object sender, System.EventArgs e)
{
    int len;
    int bytes;
    Object arrData = new byte[bytes];

    Len = axFw1.ReceiveB (arrData);
    If (len > 0)
    {
        object pBuf = new Byte[len];
        pBuf = axFw1.BinaryData;
        byte []RcvByte = (byte [])pBuf;
        .....
        .....
    }
}
```

## 2.5 Connection

### Summary

Handle to connected socket.

### Description

The Connection property specifies the handle of the current connection. This handle is not used in the Firewall ActiveX control, but can be combined with other controls to execute specific tasks.

The Connection property can only be read after a successful connection (by setting the Action property to ACTION\_CONNECT). This property cannot be written at any time.

There is no default value for this property.

### Example

Connection = Firewall.**Connection**

## 2.6 DestHost

### Summary

Remote host name.

### Description

The DestHost property specifies the name or internet address of a remote machine. This property must be set before a connection can be established (by changing the Action property). There are three possible ways of specifying a host name.

#### Machine Name

An application only needs to specify the name of the remote machine 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 remote machine 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 remote machine. The *AddrType* property must be set to FW\_ADDR\_DNS.

#### 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 remote machine. The *AddrType* property must be set to FW\_ADDR\_DNS.

#### Internet Address

Sometimes the user knows only the internet address of the remote machine 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. The *AddrType* property must be set to FW\_ADDR\_IP4.

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

```
Firewall.FirewallServer = "sparky.distinct.com"  
Firewall.FirewallPort = 1080  
Firewall.AddrType = FW_ADDR_DNS  
Firewall.DestHost = "speedy.distinct.com"  
Firewall.DestPort = 13  
Firewall.Action = ACTION_CONNECT
```

## 2.7 DestPort

### Summary

Remote port to be used for next connection.

### Description

The DestPort property specifies the port on the remote machine with which a connection should be established. Once connected, the remote port remains the same for the entire session.

The DestPort property must be set before a connection is established with the remote machine (by setting the Action property to ACTION\_CONNECT or ACTION\_ASYNC\_CONNECT). The Firewall ActiveX control does not verify the setting of the DestPort property and any value (except 0) is therefore legal.

Most TCP/IP implementations include a SERVICES file which lists the port numbers for many commonly available TCP services.

This property can be changed at any time except after a connection has been established with a remote machine (by setting the Action property to ACTION\_CONNECT or ACTION\_ASYNC\_CONNECT). There is no default value for this property.

### Example

```
Firewall.FirewallServer = "sparky.distinct.com"  
Firewall.FirewallPort = 1080  
Firewall.AddrType = FW_ADDR_DNS  
Firewall.DestHost = "speedy.distinct.com"  
Firewall.DestPort = 13  
Firewall.Action = ACTION_CONNECT
```

## 2.8 FirewallPort

### Summary

Firewall port to be used for next connection.

### Description

The FirewallPort property specifies the port on the firewall server through which a connection should be established. Once connected, the firewall port remains the same for the entire session.

The FirewallPort property must be set before a connection is established with a remote machine (by setting the Action property to ACTION\_CONNECT or ACTION\_ASYNC\_CONNECT). The Firewall ActiveX control does not verify the setting of the FirewallPort property and any value (except 0) is therefore legal.

This property can be changed at any time except after a connection has been established with a remote machine (by setting the Action property to ACTION\_CONNECT or ACTION\_ASYNC\_CONNECT). The default value for this property is 1080.

### Example

```
Firewall.FirewallServer = "sparky.distinct.com"  
Firewall.FirewallPort = 1080  
Firewall.AddrType = FW_ADDR_DNS  
Firewall.DestHost = "speedy.distinct.com"  
Firewall.DestPort = 13  
Firewall.Action = ACTION_CONNECT
```

## 2.9 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 made. 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

```
Firewall.FirewallServer = "sparky.distinct.com"  
Firewall.FirewallPort = 1080  
Firewall.AddrType = FW_ADDR_DNS  
Firewall.DestHost = "speedy.distinct.com"  
Firewall.DestPort = 13  
Firewall.Action = ACTION_CONNECT
```

## 2.10 LastResult

### Summary

Result of last action executed.

### Description

The LastResult property contains the result of the last operation executed by the Firewall ActiveX control. This property can have any one of the following values.

Value	Meaning
FW_OK	Operation completed successfully.
FW_ABORTED	Connection successfully aborted
FW_SEND_FAILED	Unable to send data.
FW_RECV_ERROR	Unable to receive data.
FW_CONNECTION_REFUSED	Firewall server refused connection.
FW_INVALID_ADDR	Invalid address is specified.
FW_NO_METHOD_SELECTED	Invalid method selected for authentication
FW_AUTH_FAILED	Authentication failed
FW_METHOD_NOTSUPPORTED	The specified authentication method is not supported
FW_UNKNOWN_VERSION	Socks version is unknown
FW_INVALID_USERNAME	The username or password is not valid
FW_TIMED_OUT	The ActiveX timed out while waiting for response from the server
FW_HOSTNAME_UNRESOLVED	Unable to resolve either the firewall host or the destination host.

The following are the possible error values when the Action is set to ACTION\_ASYNC\_CONNECT or the method AsyncConnect is called.

Value	Meaning
FW_ASYNC_SEND_FAILED	Unable to send data.
FW_ASYNC_RECV_ERROR	Unable to receive data.
FW_ASYNC_CONNECTION_REFUSED	Firewall server refused connection.
FW_ASYNC_INVALID_ADDR	Invalid address is specified.
FW_ASYNC_NO_METHOD_SELECTED	Invalid method selected for authentication
FW_ASYNC_AUTH_FAILED	Authentication failed
FW_ASYNC_METHOD_NOTSUPPORTED	The specified authentication method is not supported
FW_ASYNC_UNKNOWN_VERSION	Socks version is unknown
FW_ASYNC_INVALID_USERNAME	The username or password is not valid
FW_ASYNC_TIMED_OUT	The ActiveX timed out while waiting for response from the server
FW_ASYNC_HOSTNAME_UNRESOLVED	Unable to resolve either the firewall host or the destination host.

The LastResult property reflects the result of the last operation initiated by setting the Action or Send property.

The value of this property should be checked immediately after initiating the operation. Accessing other properties may change the value of the property.

This property can be read at any time. There is no default value for this property.

**Example**

```
Firewall.Send = "This is a test"  
If (Firewall.LastResult <> FW_OK) Then Exit Sub
```

## 2.11 Password

### Summary

A valid password .

### Description

The Password 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 (*AuthMethods*) 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.

The property does not have any default value.

### Example

```
Firewall.FirewallServer = "sparky.distinct.com"  
Firewall.FirewallPort = 1080  
Firewall.AddrType = FW_ADDR_DNS  
Firewall.DestHost = "speedy.distinct.com"  
Firewall.DestPort = 13  
Firewall.AuthMethods = "2"  
Firewall.Username = "joe"  
Firewall.Password = "distinct"  
Firewall.Action = ACTION_CONNECT
```

## 2.12 Send

### Summary

Send buffer.

### Description

The Send property is used to transfer data to the remote machine over an established connection. 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. Because of the stream nature of the TCP protocol, every assignment of data to the Send property does not necessarily correspond to a packet being sent over the network. Data may be split up over more than one packet or several buffers may be combined and sent as a single packet.

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

### Example

```
Firewall.Send = "This is a test"  
If (Firewall.LastResult <> FW_OK) Then Exit Sub
```

## 2.13 SocksVer

### Summary

The firewall server version.

### Description

The SocksVer 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 application is not sure about the socks version then it can specify both FW\_VERSION5 and FW\_VERSION4. The Distinct firewall control will automatically detect the firewall server version and make the appropriate connection.

### Example

```
Firewall.FirewallServer = "sparky.distinct.com"  
Firewall.FirewallPort = 1080  
Firewall.AddrType = FW_ADDR_DNS  
Firewall.DestHost = "speedy.distinct.com"  
Firewall.DestPort = 13  
Firewall.AuthMethods = "2"  
Firewall.Username = "joe"  
Firewall.Password = "distinct"  
Firewall.SocksVer = FW_VERSION5  
Firewall.Action = ACTION_CONNECT
```

## 2.14 Status

### Summary

Connection status.

### Description

The Status property is used to obtain the status of a connection. This property can have any one of the following values.

<b>Value</b>	<b>Meaning</b>
FW_CONNECTED	Connected to the remote host.
FW_PENDING_REPLY	Remote machine is waiting for reply.
FW_NOTCONNECTED	Not connected to the remote host.

This property should be read immediately after setting the Action property to ACTION\_SELECT or after calling the Select method. This property can be read at any time. There is no default value for this property.

### Example

```
Firewall.Action = ACTION_SELECT
If Firewall.Status = FW_NOTCONNECTED Then
    Firewall.Action = ACTION_CONNECT
End If
```

## 2.15 Username

### 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 be established with Socks version 4 server.

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

The property does not have any default value.

### Example

```
Firewall.FirewallServer = "sparky.distinct.com"  
Firewall.FirewallPort = 1080  
Firewall.AddrType = FW_ADDR_DNS  
Firewall.DestHost = "speedy.distinct.com"  
Firewall.DestPort = 13  
Firewall.AuthMethods = "2"  
Firewall.Username = "joe"  
Firewall.Password = "distinct"  
Firewall.Action = ACTION_CONNECT
```

## 2.16 UseVariant

### Summary

Send and receive binary or ascii data.

### Description

The UseVariant property is used to specify whether data is to be received or sent in binary or ASCII form. If this property is set to True, the OnReceiveB event will be fired when data arrives; otherwise the OnReceive event is fired.

This property should be set before any data arrives or before sending any data. This property can be read at any time. The default value for this property is False.

### Example

```
Firewall.UseVariant = True
```

## 3 Events

### 3.1 OnClose

#### Summary

Connection has been closed.

#### Description

The OnClose event occurs usually in response to setting the Action property to ACTION\_CLOSE (or the Close method). In some cases, the remote machine may close a connection (for example because of a long period of inactivity or because all data has been transferred). This will also trigger an OnClose event. In this case, the application must still set the Action property to ACTION\_CLOSE (or call the Close method) to free up all the resources allocated for the connection. However, this should not be done 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\_CLOSE (or calling the Close method), it will occur before the statement following the assignment of ACTION\_CLOSE to the Action property (or the call to the Close 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\_CLOSE action (or the call to the Close method) to make sure that the connection was actually terminated.

#### Example

```
Sub Firewall_OnClose ()
  If Connected = True Then
    Connected = False
    Firewall.Action = ACTION_CLOSE
  End If
End Sub
```

## 3.2 OnConnect

### Summary

Connection has been established.

### Description

The OnConnect event occurs in response to setting the Action property to ACTION\_CONNECT (or calling the Connect method) or to ACTION\_ASYNC\_CONNECT (or calling the method AsyncConnect). In case of a synchronous connect 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. If the connection is asynchronous that is the Action property has been set to ACTION\_ASYNC\_CONNECT or the AsyncConnect method has been called then the event will be fired only when the Firewall ActiveX control receives a FD\_CONNECT event.

Normally, an application should simply set a flag in response to this event. Then, this flag can be checked directly after the ACTION\_CONNECT operation (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 Firewall_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 firewall server or the remote machine cannot be established. The following describes all possible error codes delivered by this event.

Value	Meaning
ERR_FW_SERVER_NOT_DEFINED	Firewall server is not specified.
ERR_DEST_HOST_NOT_DEFINED	Remote machine is not specified.
ERR_IN_ACTION	Another action is in progress.
ERR_CANNOT_CONNECT	Cannot connect to remote machine via firewall.
ERR_CANNOT_LISTEN	Cannot open secondary connection.
ERR_CANNOT_CLOSE	Cannot close connection.
ERR_CANNOT_CHANGE_FW_PORT	Cannot change firewall port while connected.
ERR_CANNOT_CHANGE_DEST_PORT	Cannot change remote port while connected.
ERR_FW_PORT_NOT_DEFINED	Firewall port is not specified.
ERR_DEST_PORT_NOT_DEFINED	Remote port is not specified.
ERR_NOT_CONNECTED	Not connected to remote host.
ERR_CANNOT_SEND	Cannot send data.
ERR_CANNOT_BIND	Unable to bind to socket.
ERR_CONNECT_FAILED	Unable to connect to socket.

The following describes each error in more detail.

#### **ERR\_FW\_SERVER\_NOT\_DEFINED**

Firewall server has not been specified. Set the FirewallServer property before setting the Action property to ACTION\_CONNECT (or calling the Connect method) or ACTION\_LISTEN (or calling the Listen method).

#### **ERR\_DEST\_HOST\_NOT\_DEFINED**

Remote machine has not been specified. Set the DestHost property before setting the Action property to ACTION\_CONNECT (or calling the Connect method) or ACTION\_LISTEN (or calling the Listen method).

#### **ERR\_IN\_ACTION**

Another action is already in progress.

#### **ERR\_CANNOT\_CONNECT**

Cannot connect to remote machine via firewall. The FirewallServer and DestHost must be set properly or the firewall server or the remote host is down.

#### **ERR\_CANNOT\_LISTEN**

Cannot open secondary connection. The FirewallServer and DestHost must be set properly or the firewall server or the remote host is down.

**ERR\_CANNOT\_CLOSE**

Cannot close either the primary connection or the secondary connection.

**ERR\_CANNOT\_CHANGE\_FW\_PORT**

Cannot change the firewall port while connected.

**ERR\_CANNOT\_CHANGE\_DEST\_PORT**

Cannot change the remote port while connected.

**ERR\_FW\_PORT\_NOT\_DEFINED**

Firewall port has not been specified. Set the FirewallPort property before setting the Action property to ACTION\_CONNECT (or calling the Connect method) or ACTION\_LISTEN (or calling the Listen method).

**ERR\_DEST\_PORT\_NOT\_DEFINED**

Remote port has not been specified. Set the DestPort property before setting the Action property to ACTION\_CONNECT (or calling the Connect method) or ACTION\_LISTEN (or calling the Listen method).

**ERR\_NOT\_CONNECTED**

An attempt was made to set the Action property to ACTION\_SELECT (or call the Select method) without being connected.

**ERR\_CANNOT\_SEND**

Unable to send the data to the remote machine.

**ERR\_CANNOT\_BIND**

Unable to bind socket to local address.

**ERR\_CONNECT\_FAILED**

Unable to connect secondary connection to remote host.

**Example**

```
Sub Firewall_OnError (ErrorCode As Integer)
    If ErrorCode = ERR_CANNOT_SEND Then
        MsgBox "Unable to send data", 64, "Sample Program"
    End If
End Sub
```

### 3.4 OnFwListen

#### Summary

Secondary connection has been established.

#### Description

The OnFwListen event occurs in response to setting the Action property to ACTION\_LISTEN (or calling the Listen method). This event will occur before the statement following the assignment of ACTION\_LISTEN to the Action property (or the call to the Listen method) is reached.

This event delivers the internet address of the firewall server and the port the firewall server is listening on. The application should set a flag in the OnFwListen and OnError events, so that it can determine if the session has been established or not.

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

While handling the OnFwListen 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 Firewall_OnFwListen (Host As String, Port As Integer)  
    Listened = True  
End Sub
```

### 3.5 OnReceive

#### Summary

More data has been received.

#### Description

The OnReceive event occurs whenever the underlying protocol stack receives one or more bytes of additional data from the remote host. This event actually delivers the data to the application.

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 Firewall_OnReceive (Buffer As String)
    Msg.Text = Buffer
End Sub
```

## 3.6 OnReceiveB

### Summary

More data has been received and it is to be read as binary.

### Description

The OnReceiveB event occurs whenever the underlying protocol stack receives one or more bytes of additional data from the remote host. The event does not actually deliver the data to the application but instead expects that the application will call the ReceiveB method to obtain all data waiting in the buffer.

This event is fired if and only if the UseVariant property is set to True. If this property is set to False, the OnReceive event is fired instead.

While handling the OnReceiveB 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 OnReceiveB events.

### Example

```
Sub Firewall_OnReceiveB (Bytes As Long)
    Dim Buffer() As Byte
    Dim Siz As Long
    Dim i As Integer

    Siz = Firewall.ReceiveB (Buffer, Bytes)
    If Siz > 0 Then
        Open "c:\abc.exe" For Binary Access Write As #1
        For i = 1 To Siz
            Put #1, i, Buffer (x)
        Next i
        Close #1
    Else
        MsgBox "Cannot receive binary data", 64, "Sample Program"
    End If
End Sub
```

### 3.7 OnRemote

#### Summary

Connection request has been received.

#### Description

The OnRemote event occurs when the remote host is trying to make a connection to the local host. This event delivers the internet address and the port of the connecting host (remote host).

While handling the OnRemote 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 Firewall_OnRemote (Host As String, Port As Integer)
    RemoteHost = Host
    RemotePort = Port
End Sub
```

## 4. Methods

### 4.1 Abort

#### Summary

Abort current action.

#### Syntax

**Boolean Abort ()**

#### Description

The Abort method aborts any current action that is in progress. The method takes no parameter and returns a boolean. If the connection is successfully reset and closed then the method will return TRUE;; otherwise it returns FALSE. The application must ensure that the method was successfully executed by checking the return value. The method also set the LastResult property, if there is an error then the value of the LastResult can be checked to determine the nature of the error.

Calling this method is equivalent to setting the Action property to ACTION\_ABORT.

#### Example

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

## 4.2 AsyncConnect

### Summary

Asynchronously connect to remote host via a firewall.

### Syntax

#### **Boolean Connect (*FirewallServer*, *FirewallPort*, *DestHost*, *DestPort*)**

<i>FirewallServer</i>	String
<i>FirewallPort</i>	Integer
<i>DestHost</i>	String
<i>DestPort</i>	Integer

### Description

The AsyncConnect method establishes a asynchronous connection to the remote host via a firewall.

The AsyncConnect method takes a firewall server name (*FirewallServer*), a firewall port (*FirewallPort*), a remote host name (*DestHost*) and a remote port (*DestPort*) as its parameters and returns a boolean. The *FirewallServer* and *DestHost* parameters must be the name or internet address (in dotted decimal notation) of the firewall server and remote host, respectively. If the *DestHost* is the internet address then the *AddrType* property must be set to FW\_ADDR\_IP4 and if it is a machine name then the *AddrType* property must be set to FW\_ADDR\_DNS, by default the *AddrType* property has the value FW\_ADDR\_IP4. The *FirewallPort* parameter must contain the port number the firewall server is listening on. The *DestPort* parameter must contain the remote service port.

The application can also set the *AuthMethods* 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 supported currently. An authentication is only possible if the Socks version is 5, version 4 Socks server does not support any authentication protocols.

The version of the socks server can be specified in the SocksVer property, if the application leaves this property empty then the Distinct firewall control will automatically detect the Socks version.

If a connection is successfully initiated, then this method returns True; otherwise, it returns False. The application should ensure that the method was successfully executed by checking the return value. This method also sets the LastResult property. The value of the LastResult property can be checked to determine the nature of the error.

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. 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\_ASYNC\_CONNECT.

### Example

```
Firewall.AuthMethods = "2"  
Firewall.Username = "joe"  
Firewall.Password = "distinct"  
Firewall.SocksVer = FW_VERSION5  
Firewall.AddrType = FW_ADDR_DNS
```

```
Result = Firewall.AsyncConnect ("sparky.distinct.com", 1080, "speedy.distinct.com", 13)
If Result = False Then
    MsgBox "Unable to connect to remote host", 64, "Sample Program"
End If
```

## 4.3 Close

### Summary

Close connection to remote machine via firewall.

### Syntax

**Boolean Abort ()**

### Description

If a connection is no longer needed or the application wants to close an established connection, then the Close method should be called. If a secondary connection was opened, that connection will also be closed. Once the connection is closed, the OnClose event will occur before the next line of code is reached. An application must close all connections it has created before it quits.

The Close method takes no parameters and returns a boolean. If the connection is successfully 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. This method also sets the LastResult property. The value of the LastResult property can be checked to determine the nature of the error.

Calling this method is equivalent to setting the Action property to ACTION\_CLOSE.

### Example

```
Result = Fw.Close ()  
If Result = False Then  
    MsgBox "Unable to close connection", 64, "Sample Program"  
End If
```

## 4.4 Connect

### Summary

Connect to remote host via a firewall.

### Syntax

**Boolean Connect** (*FirewallServer*, *FirewallPort*, *DestHost*, *DestPort*)

<i>FirewallServer</i>	String
<i>FirewallPort</i>	Integer
<i>DestHost</i>	String
<i>DestPort</i>	Integer

### Description

The Connect method establishes a connection to the remote host via a firewall.

The Connect method takes a firewall server name (*FirewallServer*), a firewall port (*FirewallPort*), a remote host name (*DestHost*) and a remote port (*DestPort*) as its parameters and returns a boolean. The *FirewallServer* and *DestHost* parameters must be the name or internet address (in dotted decimal notation) of the firewall server and remote host, respectively. If the *DestHost* is the internet address then the *AddrType* property must be set to FW\_ADDR\_IP4 and if it is a machine name then the *AddrType* property must be set to FW\_ADDR\_DNS, by default the *AddrType* property has the value FW\_ADDR\_IP4. The *FirewallPort* parameter must contain the port number the firewall server is listening on. The *DestPort* parameter must contain the remote service port.

The application can also set the *AuthMethods* 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 supported currently. An authentication is only possible if the Socks version is 5, version 4 Socks server does not support any authentication protocols.

The version of the socks server can be specified in the *SocksVer* property, if the application leaves this property empty then the Distinct firewall control will automatically detect the Socks version.

If a connection is successfully established, then this method returns True; otherwise, it returns False. The application should ensure that the method was successfully executed by checking the return value. This method also sets the *LastResult* property. The value of the *LastResult* property can be checked to determine the nature of the error.

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

```
Firewall.AddrType = FW_ADDR_DNS
Result = Firewall.Connect ("sparky.distinct.com", 1080, "speedy.distinct.com", 13)
If Result = False Then
    MsgBox "Unable to connect to remote host", 64, "Sample Program"
End If
```

## 4.5 Listen

### Summary

Open a secondary connection.

### Syntax

**Boolean Listen (*FirewallServer*, *FirewallPort*, *DestHost*, *DestPort*)**

<i>FirewallServer</i>	String
<i>FirewallPort</i>	Integer
<i>DestHost</i>	String
<i>DestPort</i>	Integer

### Description

The Listen method establishes a secondary connection to the remote host via a firewall.

The Listen method takes a firewall server name (*FirewallServer*), a firewall port (*FirewallPort*), a remote host name (*DestHost*) and a remote port (*DestPort*) as its parameters and returns a boolean. The *FirewallServer* and *DestHost* parameters must be the name or internet address (in dotted decimal notation) of the firewall server and local host, respectively. The *FirewallPort* parameter must contain the port number the firewall server is listening on. The *DestPort* parameter must contain the local service port.

If a secondary connection is successfully established, then this method returns True; otherwise, it returns False. The application should ensure that the method was successfully executed by checking the return value. This method also sets the LastResult property. The value of the LastResult property can be checked to determine the nature of the error.

If the connection can be established, then the OnFwListen 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 Listen method) is executed. The application should set a flag in the OnFwListen 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\_LISTEN.

### Example

```
Result = Firewall.Listen ("sparky.distinct.com", 1080, "speedy.distinct.com", 13)
If Result = False Then
    MsgBox "Cannot establish a secondary connection", 64, "Sample Program"
End If
```

## 4.6 ReceiveB

### Summary

Reads binary data.

### Syntax

#### Long ReceiveB (*Buffer*, *Bytes*)

*Buffer*            Variant

*Bytes*            Long

### Description

The ReceiveB method retrieves the binary data and passes to the application.

The ReceiveB method takes a buffer (*Buffer*) and the number of bytes to read (*Bytes*) as its parameters and returns the actual number of bytes retrieved. The *Buffer* parameter should be an array of bytes data type. The *Bytes* parameter specifies how many bytes should be retrieved.

If there are less data to be read than the requested *Bytes*, only the available data are read. This is reflected in the return value. If the returned value is less than the requested *Bytes*, that means there are less data available than requested. The application should also check to make sure what is the actual number of bytes read.

### Example

```
Length = Firewall.ReceiveB (Buffer, Bytes)
```

## 4.7 Select

### Summary

Retrieve status of connection.

### Syntax

**Boolean Select ()**

### Description

While connected to the remote host, sometimes it might be necessary to know what the connection status is. The Select method can be called to find out if the connection is still open, if the remote host is waiting for a reply from the local host, or if the local host is no longer connected to the remote host.

The Select method takes no parameters and returns a boolean. If the Select method 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 Status property should be read immediately after calling the Select method to find out what the connection status is.

Calling this method is equivalent to setting the Action property to ACTION\_SELECT.

### Example

```
Result = Firewall.Select ()  
If Result = False Then  
    MsgBox "Unable to obtain connection status", 64, "Sample Program"  
End If
```

## 4.8 SendB

### Summary

Send binary data.

### Syntax

#### **Boolean SendB (*Buffer*, *Bytes*)**

*Buffer*            Variant

*Bytes*            Long

### Description

The SendB method sends the binary data over an established connection.

The SendB method takes a buffer (*Buffer*) and the number of bytes to be send (*Bytes*) as its parameters and returns a boolean. The *Buffer* parameter should be an array of bytes data type. The *Bytes* parameter must contain the number of bytes of data to send. If the data is successfully sent, then the method returns True; otherwise, it returns False. The application should ensure that the method was successfully executed by checking the return value.

### Example

```
Result = Firewall.SendB (Buffer, Bytes)
If Result = False Then
    MsgBox "Cannot send binary data", 64, "Sample Program"
End If
```



