IPX & SPX Custom Controls v2.00 For Visual Basic™ 3.0 & Windows™ 3.1

Introduction

IPX (Internetwork Packet Exchange) protocol is a datagram (connectionless) service used as the underlying protocol in Novell NetWare. IPX establishes no connections nor guarantee of delivery. Each packet a workstation sends is treated separately. SPX (Sequenced Packet Exchange) is a connection-oriented protocol providing guaranteed packet delivery, flow control, and sequencing. SPX has many of the characteristics of the transport level of OSI. IPX/SPX are adaptations of protocols developed by Xerox(r), the Xerox Network System, XNS(tm).

NetWare addressing is similar to the postal service's addressing system that identifies recipients by country, state, city, street, number, apartment, and individual names. NetWare addresses consist of three components: network number, node address, and socket. Network numbers identify each segment of an internetwork. Network numbers are assigned when NetWare file servers and bridges are installed. IPX derives the node address from the address of each network adapter card. Socket numbers provide routing within a node, since several processes can be running simultaneously. Processes can open and close sockets to receive and send packets.

The IPX and SPX custom controls allow you to communicate with remote machines connected by a local area network. To use the control, you must have the workstations set up as NetWare clients. This involves loading IPX.COM or IPXODI.COM before Windows loads. The examples provided also require installing NETX or VLMs because they use several NetWare services to find other users.

The IPX and SPX custom controls are event driven and relieve the application from polling the status of network communication channels. When new data is received from a remote machine, the IPX or SPX custom control fires a Visual Basic subroutine allowing the application to service the received information.

Several examples are provided showing how to use the controls. Further information about IPX and SPX can be found in the following references: *Client SDK Documentation* and *C Programmer's Guide* by Novell, *C Programmer's Guide To NetBIOS, IPX And SPX* by W. David Schwaderer, *Programmer's Guide to NetWare* by Charles Rose, and *Network Programming in C*, by Barry Nance.

Control Properties

(These properties apply to both controls, unless indicated otherwise)

Socket

[*form.*]Control.**Socket**[= *numericexpression*]

Sets the socket that the control listens to and sends packets out through.

Example: spx1.Socket = &H5454

ReceiveECBS

[*form.]*Control.**ReceiveECBS**[= *numericexpression*]

Settings
Default = 5

Sets the number of ECBs that are used to receive packets. Currently, this property should only be set in the design environment and not at run time.

SendECBS

[*form.]*Control.Send**ECBS**[= *numericexpression*]

Settings
Default = 1

Sets the number of ECBs that are used to send packets. Currently, this property should only be set in the design environment and not at run time. In addition, this property should currently be set to 1.

Channels

[*form*]Control.**Channels**[= *numericexpression*]
(not applicable for IPX)

Settings
Default = 1

Sets the maximum number of connections that the control will need to service. This property can be set once after the control is loaded and cannot change until the control is reloaded. Currently, this property has no effect.

Example: spx1.Channels = 1

Connection

[*form*]Control.**Connection**
(not applicable for IPX)

Returns the SPX connection ID assigned to this connection by SPX. This ID may be used to make SPX calls in NWIPXSPX.DLL.

Example: id% = spx1.Connection

Send

[*form.*]Control.**Send**[= *stringexpression*]

Sends *stringexpression* to a remote node specified by RemoteName.

Example: spx1.Send="HELLO WORLD"

Received

[*form.*]Control.**Received**

Return Data Type   **String**

Retrieves data retrieved from the network. When data is received, the control will call the Visual Basic subroutine spx1_*ReceiveData*

Example:

```
Sub spx1_ReceiveData ()
        Data$ = spx1.Received
End Sub
```

ReceivedFrom

[*form.*]Control.**ReceivedFrom**

Return Data Type   **String * 12**

Returns the internet address (network, node, and socket) of the node that sent the most recently received packet.

Example:

```
Sub spx1_ReceiveData()
        From$ = spx1.ReceivedFrom
         Data$ = spx1.Received
 End Sub
```

Event

[*form.*]Control.**Event**

Return Data Type   **Integer**

Retrieves an event about the connection. An event is any activity (received data, errors, new connection with a remote, lost connection with a remote, etc.) related to the connection. When an event occurs, the control will fire the subroutine *spx1_LinkEvent*.

Example:

```
Sub spx1_LinkEvent ()
        Event% = spx1.Event
```

End Sub

Status

[*form.*]Control.**Status** [= { 0}]
(not applicable for IPX)

Status Initialization:

0	Disable

Status Information:

1	Control is listening for a connection
2	Control is trying to establish a connection
3	Control has established a connection, ready to send
4	Control is terminating the connection

Controls a connection and will contain status information. Setting the Status property to 0 will terminate any connection.

Example: spx1.Status = 0   'Terminate any connection

LinkType

[*form.*]Control.**LinkType** [= *numericexpression*]
(not applicable for IPX)

Settings

Default	1
1	establish a session with remote
2	listen for connection

Set the control's mode of operation. This may be changed at any time.
Example: spx1.LinkType = 2

LocalName

[*form.*]Control.**LocalName**

Return Data Type   **String * 12**

Returns the internet address of the current workstation (network, node, and socket).

Example: myAddress = spx1.LocalName

RemoteName

[*form.*]Control.**RemoteName[=** *string expression***]**

Set the internet address that the control will use to send and receive data. This may be changed at any time.

Example: spx1.RemoteName = yourAddress

PacketType

[*form.*]Control.**PacketType** [= *numericexpression*]
(not applicable for SPX)

Sets the Packet Type field in the IPX header.

Settings

Default	0
0	UnKnown packet
4	IPX Packet

Example: ipx1.PacketType = 4

TaskID

[*form.*]Control.**TaskID**

Return Data Type   **Long**

Returns the task ID from NWIPXSPX.DLL for the current task.

Example: myTaskID = spx1.TaskID

Control Events

LinkEvent()

This event is called each time an event for a connection occurs. An event might be an IPX/SPX error or an event listed in the event table. The application should check the value of the Event property during this event.

Example:

```
Sub spx1_LinkEvent()
   Event% = spx1.Event()
End Sub
```

ReceiveData()

This event is called each time data is received. The application should retrieve the value of Received during this event.

Example:

```
Sub spx1_ReceiveData()
   Data$ = spx1.Received()
 End Sub
```

SentData()

This event is called each time the control has transmitted data.

Example:

```
Sub spx1_SentData()
     ' Ready to send next packet
 End Sub
```

Example Applications

NETCHAT.MAK

This example application uses the IPX control to send IPX packets to another workstation and allow the users to chat to one another. The application uses a simple protocol and finite state machine to connect to another application, send and receive data, and then disconnect from the other application. This application also uses several calls provided by NWCALLS.DLL to find all the other users and their internet addresses.

CLIENT.MAK & SERVER.MAK

These two examples demonstrate the SPX control in a simple client/server protocol. The client application sends a directory command to the server, which returns a directory listing of the server's current directory. To set an application, all that is needed is to set the LinkType property to 2. To be a client, set the LinkType property to 1.

Event Values

Any of the following events will trigger a *spx1_LinkEvent* call. The value returned by Event is listed in the left column and its meaning is listed on the right.

237	Abnormal Connection Termination
238	Invalid Connection
239	Connection Table Full
241	IPX/SPX Not Installed
242	No DOS Memory
243	No Free ECB
244	Lock Failed
245	Over the Maximum Limit
246	IPX/SPX Previously Initialized
252	Socket Not Opened SPX Command Canceled with IPXCancelEvent

253	Malformed Packet SPX Packet Overflow
255	SPX Socket Not Opened

Visual Basic Errors

The following is a list of special errors than can be returned by the IPX/SPX custom controls. The value in the left column is the error code returned in the Visual Basic "Err" variable and its meaning is on the right.

22001	IPX not installed
22002	SPX not installed
22003	Currently only 1 user allowed
22004	Socket not opened