DELPHI SOCKETS COMPONENT V2.0

Written by: Gary T. Desrosiers
desrosi@pcnet.com
71062,2754

SOCKETS component provides Delphi with a component capable of performing TCP/IP Socket's functions by interfacing with WINSOCK.DLL.

To use this component you must have a WinSock 1.1 compatable TCP/IP stack. The control has been tested with Trumpet 2.1C, Chameleon 4.03, PC/TCP, and the native stacks in Windows NT and Windows 95.

This is Version 2.0 of the component. Most improvements and modifications have been made from suggestions that I received. One of the biggest compliaints has been the lack of good examples. To correct this, I have written an FTP client that uses many of the new features of this component and also a very general program that shows how to handle both client and server requests as well as multiple concurrent clients going to a single server. The other most requested feature was the ability to send and receive data that is greater that the length of a Pascal string. This was implemented using the custom methods; SSend and SReceive that accept PChar instead of Pascal style strings.

The following is a summary of the Version 2.0 improvements;

- Added properties;
  - MasterSocket, Gets the listener's socket
  - Peek, Preview data in the input buffer.
  - NonBlocking, Blocking vs Non-Blocking sockets
  - Timeout, For blocking mode timeouts
  - OOB, Sends and receives data out of band (urgent data)

- Modified properties;
  - SocketNumber to read/write
  - Text (no longer published)

- Added Methods;
  - SCancelListen, new method cancels the listener socket
  - GetPeerIPAddr, returns partners IP address
  - GetPeerPort, returns partners port

- Modified Methods;
  - GetIPAddr, Documented and bug fix
  - GetPort, Documented
  - SClose, Added shutdown, etc.
  - SReceive, Modified to use PChar instead of Pascal strings
  - SSend, Modified to use PChar instead of Pascal strings
  - SetText, Now loops until entire buffer sent

- Added Events
  - OnErrorOccurred, Called on WinSock errors.

- Example programs
  - cltsvr.zip contains a very basic client/server program that shows how to use this component.

- ftp.zip contains an FTP client
- rexec.zip implements a basic rexec client

Files contained in SOCKV2.ZIP

SOCKETS.PAS	- The SOCKETS component
SOCKETS.DCR	- Component resource file
SOCKETS.WRI	- What you're reading now.
REXEC.ZIP	- REXEC client sample application
FTP.ZIP	- FTP client sample application
CLTSVR.ZIP	- Simple client/server example application

Installation

First, make sure that you have WinSock.DLL and that it's avaialble and functional.
Then...
In Delphi, drop down the Options + Install Components... menu.
Click on Add...
Enter the path and file name of where the SOCKETS.PAS module is located.
Click on OK
The SOCKETS component should now appear in the Samples folder within Delphi.
Test it out using one of the supplied applications.

Summary of the implemented properties

Property Name	Writable	Readable	Design time	Run time
IPAddr	yes	yes	yes	yes
Port	yes	yes	yes	yes
SocketNumber	yes	yes	no	yes
MasterSocket	yes	yes	no	yes
Text	yes	yes	no	yes
Peek	no	yes	no	yes
OOB	yes	yes	no	yes
NonBlocking	yes	yes	yes	yes
Timeout	yes	yes	yes	yes

Summary of the implemented Methods

SConnect	Connect to listening server
SListen	Listen on Port
SCancelListen	Cancel listen request
SAccept	Accept client connection
SClose	Close sockets
SReceive	Receive PChar data
SSend	Send PChar data
GetPort	Get local port of SocketNumber
GetIPAddr	Get local IP Address
GetPeerPort	Get partner's port assignment
GetPeerIPAddr	Get partner's IP Address

Summary of the implemented Events

OnDataAvailable	Called when data is available to be received on the socket
OnSessionAvailable	Called when a session is available to be accepted
OnSessionClosed	Called when a connection is lost
OnSessionConnected	Called when an SConnect completes
OnErrorOccurred	Called on error conditions

Implemented Properties

IPAddr

Sets the IP Address of the partner that you will eventually SConnect to. You may specify this as dotted decimal or a literal name to be converted via DNS.
examples;

```
Sockets1.IPAddr := 'desrosi';
Sockets1.IPAddr := '127.0.0.1';
addr := Sockets1.IPAddr;
```

Port

Sets the Port number of the remote port to connect to or the local port to listen on depending on whether you subsequently issue a SConnect or SListen. This can be specified as a number or a literal name to be converted via DNS.
examples;

```
Sockets1.Port := 'echo';
Sockets1.Port := '7';
port := Sockets1.Port;
```

SocketNumber

Returns (or sets) the socket number of the currently allocated connection.
example;

```
sock := Sockets1.SocketNumber;
```

MasterSocket

Returns (or sets) the master socket number (listener)
example;

```
msock := Sockets1.MasterSocket;
```

Text

if set, sends the text to the partner.
if read, receives some text from the partner.
examples;

```
buffer := Sockets1.Text; (* Receive data *)
Sockets1.Text := 'This is a test'; (* Send Data *)
```

Peek

Returns up to 255 characters of data waiting to be received but does not actually receive the data.
example;

```
buffer := Sockets1.Peek; (* Look at arriving data *)
```

OOB

if set, sends the text to the partner as urgent (out of band) data.
if read, receives urgent (out of band) data.
examples;

```
buffer := Sockets1.OOB;
Sockets1.OOB := 'ABOR'; (* Send abort urgently *)
```

NonBlocking

Set to False for blocking mode and True for non-blocking mode (the default). When the socket is in blocking mode, none of the event callback functions (with the exception of OnErrorOccurred) will function.

Timeout

When NonBlocking = 0 (blocking mode) this value specifies the maximum amount of time that a socket operation can take. After this time limit expires, the operation is canceled and an error occurs. The default is 30 (seconds). The Valid range is 0-60 seconds. Setting Timeout to zero causes the operation to wait indefinitely.

Implemented Methods

SConnect

Connects to the remote (or local) system specified in the IPAddr and Port properties.
example;

```
Sockets1.SConnect; (* Connect to partner *)
```

SListen

Listens on the port specified in the Port property.
example;

```
Sockets1.SListen; (* Establish server environment *)
```

SCancelListen

Cancels listens on the socket.
example;

```
Sockets1.SCancelListen; (* Dont accept further clients *)
```

SAccept

Accepts a client request. Usually issued in OnSessionAvailable event.
example;

```
Sock := Sockets1.SAccept; (* Get client connection *)
```

SClose

Closes the socket.
example;

```
Sockets1.SClose; (* Close connection *)
```

SReceive

Receives data from partner, similar to

reading the property Text although this function
uses PChar instead of Pascal strings.
example;

```
len := Sockets1.SReceive(Sockets1.SocketNumber,szBuffer,4096);
```

SSend

Sends data to the partner, similar to
setting the property Text although this function
uses PChar instead of Pascal strings.
example;

```
len := Sockets1.SSend(Sockets1.SocketNumber,szBuff,32000);
```

GetPort

Returns the actual port number of the socket
specified as the argument. Generally used when you've
specified a port of zero and need to retrieve the
assigned port number.

GetIPAddr

Returns the IP Address of the socket specified
as the argument.

GetPeerPort

Returns the partners port number of the socket
specified as the argument.

GetPeerIPAddr

Returns partners IP Address of the socket
specified as the argument.

Implemented Events

OnDataAvailable

Called when data is available to
be received from the partner. You should issue;

```
buffer := Sockets1.Text;
```

or a SReceive method to receive the data from the partner.

OnSessionAvailable

Called when a client has requested
to connect to a 'listening' server. You can call
the method SAccept here.

OnSessionClosed

Called when the partner has closed
a socket on you. Normally, you would close your side
of the socket when this event happens.

OnSessionConnected

Called when the SConnect has
completed and the session is connected. This is a
good place to send the initial data of a conversation.
Also, you may want to enable certain controls that

allow the user to send data on the conversation here.

OnErrorOccurred

Called when an error occurs on the socket.
If defined, the OnErrorOccurred procedure is called when
the error occurs. If the procedure isn't defined then
a dialog box is displayed with the error text and the
program is halted.

I respond to all e-mail sent to me concerning this component and will (to the best of my ability) try and solve any problems and answer any question as long as you are not using this code for commercial purposes. All code including the example programs are released to the public domain and as such can be used in any mannor you see fit.

Gary T. Desrosiers
desrosi@pcnet.com
compuserve: 71062,2754