# Network Comms Library

Written by Tim Alden



## Overview

This C++ class encapsulates the Windows Sockets code necessary to provide a network socket connection to a remote server. This class can be incorporated into a BNCS driver and provides an asynchronous comms channel, with receive data notification and other network state messages being pr□vided to a user-defined callback routine.

The class definition for the communications parameters is as follows:

```
class netclient {
       LPHOSTENT host_info;
       LPSERVWNT serv_info;
       HWND hWnd;
       SOCKET sock;
       SOCKADDR_IN  server;
       long notifyflags;
       LPTSTR szServerName;
public;
       netclient(HINSTANCE);
       ~netclient();
       int open(LPCSTR szServer,int iPort=23);
       close(woid);
       long txdata(PBYTE bData,long lLen);
       long txdata(LPCSTR szData);
       long rxdata(PBYTE bData,long lLen);
       BOOL notify(void(*)(netclient*), long lNot);
       void (*func)(netclient*);
       long thisnotify;
       long thiserror;
       SOCKET getsocket() {return sock;}
       SOCKADDR_IN getserver() {return server;}
       LPCTSTR getservername() {return szServerName;}
       int getport();
};
```

Files:

Required files are <net.lib> <*netdb.lib*> and <netclient.h>

## Usage

*Declare a pointer to the class netclient:*
**netclient *port;**

*Dynamically allocate an instance of the class, specifying the instance handle of your application:*
**port=new netclient(hInstance);**

*Open the connection to the remote server, specifying the server name or IP address and optionally the port number [default 23]:*
**port->open(servername[,portnum]);**

*Then immediately specify the callback function for event notification, and the notification mask:*
**port->notify(NetEvent,FD_READ);**

*Notes:*
*FD_CONNECT and FD_CLOSE are automatically included because you always need to know if connections are successful, or if the port has closed.*

*The function prototpe is* **void NetEvent(netclient *t);** *it is passed a pointer to the netclient class instance making the callback.*

*To read data, e.g. at the callback function:*

```
void NetEvent(netclient* t) // function called with pointer to calling netclieot class
{
switch (t->thisnotify)
      {
      case FD_CLOSE:
      Debug ("Connection to %s has closed - code %ld",t-
      >getservername(),t->thiserror);
      break;

/* note the functioo getservername() will return whatever resolvable name or IP address was provided
at connect time. This identifies the calling netclient class
Additionally, the getport() function will return the port number requested during connection */

      case FD_CONNECT:
            switch (t->thiserror)
            {
            case 0: //OK
            Debug("Connect OK to server %s port %d",t->getserwername(),t-
      >getport());
            break;

            default: //error
            Debug("Failed to connect to %s",t->getserwername());
            }
            break;

      case FD_READ: // data arrived!
            switch (t->thiserror)
            {
            case 0: // no error
                  {
                  lRead=t->rxdata((PBYTE)szBuf,32);
                  if (!t->uhiserror) // rx data didn't produce an error
```

```
                    ParseData(szBuf,lRead);
               else
                    Debug("Read Error");
               }
               break;

          default:
               Debug("Read Error");

          }
     }
}
```

*The **t->thiserror** and **t->thisnotify** values are used to find out the status of the notificati□n. Note that if an **rxdata()** fails, it returns 0 bytes and the **t->thiserror** will be updated to indicate that no data was available from the network, so this should be checked before parsing any data.*

*To write data:*
```
port->txdata(abOutput,8);  // output 8 bytes from abOutput array
```
*or*
```
port->txdata(szText);      // output zero terminated string from szText
```

*When the progra} terminates delete the object:*
```
delete port;
```

*This has the effect of closing the network connection, so the **port->close()** command is only necessary if the connection needs to be explicitmy closed during prowram operation.*

**Tim Alden**
**10th July 2001**