**Asynchronous Client Socket Example**

**.NET Framework 4**

The following example program creates a client that connects to a server. The client is built with an asynchronous socket, so execution of the client application is not suspended while the server returns a response. The application sends a string to the server and then displays the string returned by the server on the console.

**C#**

using System;

using System.Net;

using System.Net.Sockets;

using System.Threading;

using System.Text;

// State object for receiving data from remote device.

public class StateObject {

// Client socket.

public Socket workSocket = null;

// Size of receive buffer.

public const int BufferSize = 256;

// Receive buffer.

public byte[] buffer = new byte[BufferSize];

// Received data string.

public StringBuilder sb = new StringBuilder();

}

public class AsynchronousClient {

// The port number for the remote device.

private const int port = 11000;

// ManualResetEvent instances signal completion.

private static ManualResetEvent connectDone =

new ManualResetEvent(false);

private static ManualResetEvent sendDone =

new ManualResetEvent(false);

private static ManualResetEvent receiveDone =

new ManualResetEvent(false);

// The response from the remote device.

private static String response = String.Empty;

private static void StartClient() {

// Connect to a remote device.

try {

// Establish the remote endpoint for the socket.

// The name of the

// remote device is "host.contoso.com".

IPHostEntry ipHostInfo = Dns.Resolve("host.contoso.com");

IPAddress ipAddress = ipHostInfo.AddressList[0];

IPEndPoint remoteEP = new IPEndPoint(ipAddress, port);

// Create a TCP/IP socket.

Socket client = new Socket(AddressFamily.InterNetwork,

SocketType.Stream, ProtocolType.Tcp);

// Connect to the remote endpoint.

client.BeginConnect( remoteEP,

new AsyncCallback(ConnectCallback), client);

connectDone.WaitOne();

// Send test data to the remote device.

Send(client,"This is a test<EOF>");

sendDone.WaitOne();

// Receive the response from the remote device.

Receive(client);

receiveDone.WaitOne();

// Write the response to the console.

Console.WriteLine("Response received : {0}", response);

// Release the socket.

client.Shutdown(SocketShutdown.Both);

client.Close();

} catch (Exception e) {

Console.WriteLine(e.ToString());

}

}

private static void ConnectCallback(IAsyncResult ar) {

try {

// Retrieve the socket from the state object.

Socket client = (Socket) ar.AsyncState;

// Complete the connection.

client.EndConnect(ar);

Console.WriteLine("Socket connected to {0}",

client.RemoteEndPoint.ToString());

// Signal that the connection has been made.

connectDone.Set();

} catch (Exception e) {

Console.WriteLine(e.ToString());

}

}

private static void Receive(Socket client) {

try {

// Create the state object.

StateObject state = new StateObject();

state.workSocket = client;

// Begin receiving the data from the remote device.

client.BeginReceive( state.buffer, 0, StateObject.BufferSize, 0,

new AsyncCallback(ReceiveCallback), state);

} catch (Exception e) {

Console.WriteLine(e.ToString());

}

}

private static void ReceiveCallback( IAsyncResult ar ) {

try {

// Retrieve the state object and the client socket

// from the asynchronous state object.

StateObject state = (StateObject) ar.AsyncState;

Socket client = state.workSocket;

// Read data from the remote device.

int bytesRead = client.EndReceive(ar);

if (bytesRead > 0) {

// There might be more data, so store the data received so far.

state.sb.Append(Encoding.ASCII.GetString(state.buffer,0,bytesRead));

// Get the rest of the data.

client.BeginReceive(state.buffer,0,StateObject.BufferSize,0,

new AsyncCallback(ReceiveCallback), state);

} else {

// All the data has arrived; put it in response.

if (state.sb.Length > 1) {

response = state.sb.ToString();

}

// Signal that all bytes have been received.

receiveDone.Set();

}

} catch (Exception e) {

Console.WriteLine(e.ToString());

}

}

private static void Send(Socket client, String data) {

// Convert the string data to byte data using ASCII encoding.

byte[] byteData = Encoding.ASCII.GetBytes(data);

// Begin sending the data to the remote device.

client.BeginSend(byteData, 0, byteData.Length, 0,

new AsyncCallback(SendCallback), client);

}

private static void SendCallback(IAsyncResult ar) {

try {

// Retrieve the socket from the state object.

Socket client = (Socket) ar.AsyncState;

// Complete sending the data to the remote device.

int bytesSent = client.EndSend(ar);

Console.WriteLine("Sent {0} bytes to server.", bytesSent);

// Signal that all bytes have been sent.

sendDone.Set();

} catch (Exception e) {

Console.WriteLine(e.ToString());

}

}

public static int Main(String[] args) {

StartClient();

return 0;

}

}