How to Use NetworkStream in Windows Applications

NetworkStream is a fundamental class in the .NET framework that provides the underlying stream for network access. It is essential for developers working on networked applications where data needs to be sent or received over TCP/IP connections. In the Windows environment, NetworkStream is particularly useful for creating client-server applications, enabling efficient data transmission between networked devices.

NetworkStream is part of the System.Net.Sockets namespace and is used in conjunction with the TcpClient and TcpListener classes. This article will guide you through the process of creating a simple client-server application using NetworkStream in a Windows environment.

Examples:

1. **Creating a TCP Server Using NetworkStream:**

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;

class TcpServer
{
    public static async Task StartServer()
    {
        TcpListener listener = new TcpListener(IPAddress.Any, 8080);
        listener.Start();
        Console.WriteLine("Server started, waiting for connection...");

        while (true)
        {
            TcpClient client = await listener.AcceptTcpClientAsync();
            Console.WriteLine("Client connected!");

            NetworkStream stream = client.GetStream();
            byte[] buffer = new byte[1024];
            int bytesRead = await stream.ReadAsync(buffer, 0, buffer.Lengt
h);
            string message = Encoding.UTF8.GetString(buffer, 0, bytesRead)
;
            Console.WriteLine($"Received: {message}");

            byte[] response = Encoding.UTF8.GetBytes("Message received");
```

```
            await stream.WriteAsync(response, 0, response.Length);
            client.Close();
        }
    }

    static void Main(string[] args)
    {
        Task.Run(() => StartServer()).Wait();
    }
}
```

2. **Creating a TCP Client Using NetworkStream:**

```
using System;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;

class TcpClientApp
{
    public static async Task StartClient()
    {
        TcpClient client = new TcpClient();
        await client.ConnectAsync("127.0.0.1", 8080);
        Console.WriteLine("Connected to server!");

        NetworkStream stream = client.GetStream();
        byte[] message = Encoding.UTF8.GetBytes("Hello, Server!");
        await stream.WriteAsync(message, 0, message.Length);

        byte[] buffer = new byte[1024];
        int bytesRead = await stream.ReadAsync(buffer, 0, buffer.Length);
        string response = Encoding.UTF8.GetString(buffer, 0, bytesRead);
        Console.WriteLine($"Received from server: {response}");

        client.Close();
    }

    static void Main(string[] args)
    {
        Task.Run(() => StartClient()).Wait();
    }
}
```

These examples demonstrate how to create a simple TCP server and client using NetworkStream in a Windows environment. The server listens for incoming connections and reads data from the client, while the client connects to the server and sends a message.