C# WebSocket Server Documentation

Author: Dennis “MazyModz” Andersson

Note - This documentation assumes you have decent knowledge of the C# programming language (especially within socket programming) and the WebSocket framework.

TABLE OF CONTENT

Introduction *1*

C# WebSocket Server is a easy and user-friendly Listen server for WebSocket clients written in Microsoft’s C# language. The code itself is located in 3 class files - Server.cs, Client.cs and Helpers.cs. A listen server is easily created by simply creating a new Server object. After the object is created, you will have to bind the events that the server object has where you choose what to do with the data.

Note - This System currently does not support WebSocket binary frames, only text data frames. However, you will most likely only use text data.

Server.cs *2*

This is the object for all listen servers. When you create a new server you will have to provide an IPEndPoint which will be used by the server to listen to. When you create a new server object it immediately starts to listen for incoming connections on the given IPEndPoint.

When an incoming connection is detected *connectionCallback()* is called. There the server will get the WebSocket http upgrade handshake key, hash it, and send the http response protocol to the client that wants to connect. Afterwards the a client object is created for the connected client which is then added to the list of connected clients. Then the *OnClientConnected* event is called passing the create client object as its event args.

When a message is received from a client, the *OnMessageReceived* event is called passing the client object that sent the message and the message as event args.

When a message is received from a client, the *OnClientDisconnected* event is called passing the client object that disconnected as event args.

When the server sends a message to the client, the *OnSendMessage* event is called passing the client object that the message was sent to and the message as event args.

Client.cs *3*

This is the object that is created by the server for a accepted connection. Its has a server object field which is the server that the client is connected to. it also has a socket field which is the socket of the client.

When a client object is created, a server object and socket object must be passed. On creation the client immediately starts to asynchronously wait for data to be received.

When data is received, the *messageCallback*() method is called. There, the data opcode is retrieved from the websocket data frame. If the opcode indicates that the user has disconnected, the server *OnClientDisconnected* event is called when the current client as event args, and the method is retuned. However, if the opcode indicates that a text data was received, it calls the server *OnMessageReceived* event with the current client and decoded message as event args. After the event is called, we again asynchronously wait for data be received.

Installation/ usage *4*

Firstly, make sure you have the following three files added to your project: Server.cs, Client.cs and Helpers.cs.

To create a listen server, simply create a new server object. Example:

```
// Console application entry point
static void Main(string[] args)
{
    // Create a new websocket server
    Server server = new Server(new IPEndPoint(IPAdress.Parse(“127.0.0.1”), 80));
}
```

After we have created a new server object we need to bind the server events. The server object have the following events;

- OnClientConnected (EventArgs: OnClientConnectedHandler)
- OnClientDisconnected (EventArgs: OnClientDisconnectedHandler)
- OnMessageReceived (EventArgs: OnMessageReceivedHandler)
- OnSendMessage (EventArgs: OnSendMessageHandler)

We must bind these events where we created the server object. So in this example, the console application main function. We can bind these events with an anonymous function or a callback method. Example:

```
// Console application entry point
static void Main(string[] args)
{
    // Create a new websocket server
    Server server = new Server(new IPEndPoint(IPAdress.Parse(“127.0.0.1”), 80));

    // Bind the event for when a client connected with lambda
    server.OnClientConnected += (object sender, OnClientConnectedHandler e) =>
    {
        string clientGuid = e.GetClient().GetGuid();
        Console.WriteLine(“Client with guid {0} connected!”, clientGuid);
    };

    // Bind the event for when a message is received with a callback method
    server.OnMessageReceived += MessageReceivedCallback;

}

// Callback method for when a message was received
static void MessageReceivedCallback(object sender, OnMessageReceivedHandler e)
{
    string message = e.GetMessage();
    string guid = e.GetClient().GetGuid();
    Console.WriteLine(“Message received: {0} - Sender Guid: {1}”, message, guid);
}
```

That’s all there is to it for the server. Next up we have to setup the client on the websocket. To do this we need to create a javascript websocket object and bind it to the same ip and port as what you did when you created the server object. Then we have to bind the websocket events. Example:

```
<script>

    // Create new websocket client to connect to our server
    var socket = new WebSocket(“ws://127.0.0.1:80”);

    socket.onopen = function(e) {
        alert(‘Connected to the server!’);
    }

    socket.onclose = function(e) {
        alert(‘Disconnected from the server!’);
    }

    socket.onmessage = function(e) {
        alert(‘Server says: ’+e.data);
    }

    socket.onerror = function(e) {
        alert(‘There was an error!’);
    }

</script>
```

That’s all there is to it!