# Developer Manual

**jWebSocket**

**C# Client Library**

**Version 1.0**

# Version Control

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 05/13/12 | 1.0 | Creation of document | Rolando Betancourt Toucet |

## Overview

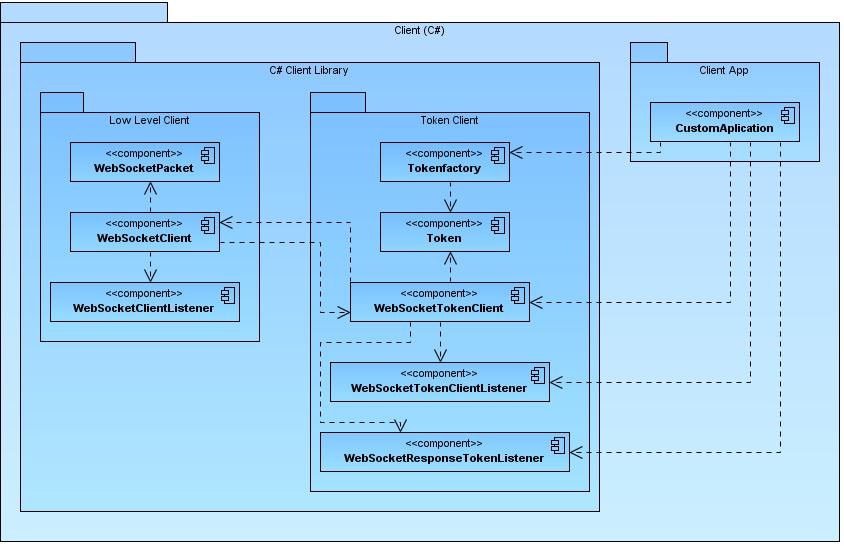
The C # Client Library is only a WebSocket protocol implementation in the C# programming language that provides a well-defined API through which to achieve integration between client applications developed with this language and framework jWebSocket. Using this library allows developers to focus entirely on the application logic as well as a considerable saving of time in implementing them.

## Infrastructure, Model, Focusing

In developing this library was followed a layered architecture which offers the possibility to divide the implementation into two main layers. A low-level layer which handles the connection and data transfer to any server to use the WebSocket protocol, and other high-level layer that meets the specifications of the server jWebSocket to achieve communication with itself

With the objective of reuse as much code as possible we used a component model for developing the various features of the library. This brings greater organization and independence of the code.

The following image shows a component diagram to help understand better the solution.

**

**Picture 1:** [Components Diagram of the C# Client Library](Components%20Diagram.jpg)

**Design Pattern**

Because this library is responsible for sending and receiving information all the time is convenient to use the Observer pattern for greater fluidity and notification of events. This pattern resolves the need to maintain consistency between related objects without requiring that classes are highly coupled.

**Technologies Used**

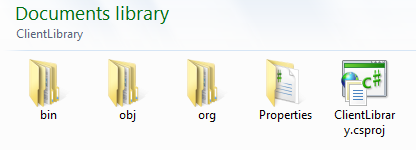
For development we used the programming language C # with. NET Framework 4.0 and the integrated development environment (IDE) Visual Studio 2010

## Requirements and prerequisites

Are not necessary

## Modules, Structure

|  |  |
| --- | --- |
| **Project Name** | C# Client Library |
| **Location of sources** | *<https://jwsdev.org:9443/svn/jWebSocket/branches/jWebSocket-1.0/jWebSocketCSClient>* |
| **SVN branch** | jWebSocket-1.0 |
| **File .zip** | SourceCodeCSharpClientLibrary.zip |
| **Directory Structure** |  |
| ***org.jwebsocket.client.common:***  It contains all the implementations that are used by the low level client and token client | |
| ***org.jwebsocket.client.csharp.api:***  It contains all the interfaces that are implemented by the low level client. | |
| ***org.jwebsocket.client.csharp.cbase:***  Contains the implementation of the base classes for low level client. | |
| ***org.jwebsocket.client.csharp.kit:***  Contains the implementation of the tools used by the low level client. | |
| ***org.jwebsocket.client.token.api:***  It contains all the interfaces that are implemented by the Token client. | |
| ***org.jwebsocket.client.token.kit:***  Contains the implementation of the tools used by the client token. | |
| ***org.jwebsocket.client.token.processor:***  Contains the implementation relating to the processing of a Token. | |
| ***org.jwebsocket.client.token.tbase:***  Contains the implementation of the base classes for Token client. | |

****

***Picture 2:*** *Directory*

**bin:** This directory stores the compiled source code.

**obj:** This directory temporarily stores the compiled source code, its content is not included in the version control.

**org:** This directory contains all the source code of classes and libraries of the solution.

**Properties:** This directory contains the data dl solution assembly; its content is not included in version control.

## Source Code

|  |  |  |
| --- | --- | --- |
| **Packet** | Low Level Client | |
| **Component** | WebSocketClient | |
| **Class** | WebSocketBaseClient.cs | |
| **Extend** | WebSocketClient.cs | |
| **Description** | This class implements the protocol WebSocket | |
| **Dependencies** | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Net.Sockets;  using System.Net.Security;  using System.Security.Authentication;  using System.Security.Cryptography.X509Certificates;  using System.IO;  using System.Threading;  using System.Diagnostics;  using log4net;  using log4net.Config;  using ClientLibrary.org.jwebsocket.client.csharp.api;  using ClientLibrary.org.jwebsocket.client.csharp.kit;  using ClientLibrary.org.jwebsocket.client.common; | |
| **Attributes** | | |
| mURI | Uri**:** Server URI | |
| mSocket | TcpClient: Provides an API to connect to a server application through a TCP channel. | |
| mNetStream | NetworkStream: allows a connection through a socket using TcpClient. | |
| mVersion | Int: WebSocket protocol version to be used by default in connection with the server. | |
| mListeners | List<WebSocketClientListener>: listeners list of the application. | |
| mSubProtocols | List<WebSocketSubProtocol>: List of sub-protocols used to connect to the server. | |
| mNegotiatedSubProtocol | WebSocketSubProtocol: Protocol that is negotiated with the server to connect. | |
| mEncoding | WebSocketEncoding: Default encoding | |
| mStatus | WebSocketStatus: Status of the connection to the server. | |
| mReliabilityOptions | WebSocketReliabilityOptions: Reliability options to establish the connection. | |
| mHeaders | WebSocketHeaders: Contains the request and response headers. | |
| mCLose | WebSocketCloseReason: Reason to close the application. | |
| mIsRunning | Bool: State of execution of the application. | |
| **Methods** | | |
| public WebSocketBaseClient() | | Class constructor |
| public void Open(string aURI) | | Establishes a connection to the given server. |
| public void SendText(string aUTF8String) | | Send a text token to the server. |
| public void SendBinary(byte[] aBinaryData) | | Send a binary token to the server. |
| public void SendText(string aUTF8String, int aFragmentSize) | | Send a fragment token text to the server. |
| public void SendBinary(byte[] aBinaryData, int aFragmentSize) | | Send a fragment token binary to the server. |
| public void Close() | | Finish the connection to the server. |
| public bool IsRunning() | | Determines if the application is running. |
| public Dictionary<string, string> GetRequestHeader() | | Gets the header of the request. |
| public Dictionary<string, string> GetResponseHeader() | | Gets the header of the response. |
| public void AddListener(WebSocketClientListener aListener) | | Adds a listener to the application. |
| public void RemoveListener(WebSocketClientListener aListener) | | Remove a listener to the application. |
| public void OnOpen(WebSocketHeaders aHeader) | | Open callback |
| public void OnClose(WebSocketCloseReason aCloseReason) | | Close callback |
| public void OnError(WebSocketError aError) | | Error callback |
| public virtual void OnTextMessage(WebSocketPacket aDataPacket) | | Text message callback |
| public void OnBinaryMessage(WebSocketPacket aDataPacket) | | Binary message callback |
| public void OnPing() | | Ping callback |

|  |  |  |
| --- | --- | --- |
| **Packet** | Low Level Client | |
| **Component** | WebSocketClientListener | |
| **Class** | WebSocketClientListener.cs | |
| **Description** | Listener to client base | |
| **Dependencies** | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using ClientLibrary.org.jwebsocket.client.csharp.kit; | |
| **Methods** | | |
| void ProcessOnTextMessage(WebSocketPacket aDataPacket) | | Process text message callback. |
| void ProcessOnBinaryMessage(WebSocketPacket aDataPacket) | | Process binary message callback. |
| void ProcessOnFragment(WebSocketPacket aFragment, int aIndex, int aTotal) | | Process fragment callback. |
| void ProcessOnOpen(WebSocketHeaders aHeader) | | Process open callback. |
| void ProcessOnClose(WebSocketCloseReason aCloseReason) | | Process close callback. |
| void ProcessOnError(WebSocketError aError) | | Process error callback. |
| void ProcessOnPing() | | Process ping callback. |
| void ProcessOnPong() | | Process pong callback. |

|  |  |  |
| --- | --- | --- |
| **Packet** | Low Level Client | |
| **Component** | WebSocketPacket | |
| **Class** | WebSocketRawPacket.cs | |
| **Extend** | WebSocketPacket.cs | |
| **Description** | Implements the low level data packets which are interchanged between client and server | |
| **Dependencies** | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using ClientLibrary.org.jwebsocket.client.csharp.api;  using ClientLibrary.org.jwebsocket.client.csharp.kit;  using ClientLibrary.org.jwebsocket.client.common; | |
| **Attributes** | | |
| mByteArray | byte[]: Contains the package information. | |
| mFragments | string[]: Containing fragments package. | |
| mFragmentsLoaded | Int: Number of charged fragments. | |
| mFragmentsExpected | Int: Number of expected fragments. | |
| mIsFragmented | Bool: True if the packet is a fragment. | |
| mIsComplete | Bool: True if complete fragmentation. | |
| mCreationDate | DateTime: Package creation date. | |
| mTimeout | Long: Timeout packet | |
| mFrameType | WebSocketFrameType: Frame type package. | |
| **Methods** | | |
| public WebSocketRawPacket(int aInitialSize) | | Builds a packet with an initial size. |
| public WebSocketRawPacket(byte[] aByteArray) | | Building a package with an array of data. |
| public WebSocketRawPacket(WebSocketFrameType aFrameType, byte[] aByteArray) | | Building a package with an array of data and type of frame. |
| public WebSocketRawPacket(string aString) | | Building a package with a string data. |
| public WebSocketRawPacket(WebSocketFrameType aFrameType, string aString) | | Building a package with a string data and type of frame. |
| public WebSocketRawPacket(string aString, WebSocketTypeEncoding aEncoding) | | Building a package with a string data and type of encoding. |
| public void InitFragmented(int aTotal) | | Fragment the packet in a given amount. |
| public void SetFragment(string aString, int aIdx) | | Assigns a fragment to a given position. |
| public void SetString(string aString) | | Set string data packet. |
| public void SetString(string aString, WebSocketTypeEncoding aEncoding) | | Set string data packet whit type encoding. |
| public void SetUTF8(string aString) | | Set UTF8 string data. |
| public void SetASCII(string aString) | | Set ASCII string data. |
| public string GetString() | | Get string data. |
| public string GetString(WebSocketTypeEncoding aEncoding) | | Get string data with type encoding. |
| public string GetUTF8() | | Get UTF8 string data. |
| public string GetASCII() | | Get ASCII string data. |

|  |  |  |
| --- | --- | --- |
| **Packet** | Token Client | |
| **Component** | WebSocketTokenClient | |
| **Class** | WebSocketBaseTokenClient.cs | |
| **Extend** | WebSocketBaseClient.cs | |
| **Description** | Token based implementation of WebSocketBaseClient. | |
| **Dependencies** | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using ClientLibrary.org.jwebsocket.client.token.kit;  using ClientLibrary.org.jwebsocket.client.csharp.csbase;  using ClientLibrary.org.jwebsocket.client.csharp.kit;  using ClientLibrary.org.jwebsocket.client.csharp.api;  using ClientLibrary.org.jwebsocket.client.token.api;  using ClientLibrary.org.jwebsocket.client.common;  using ClientLibrary.org.jwebsocket.client.token.processor; | |
| **Attributes** | | |
| CURRENT\_TOKEN\_ID | Int: Current token ID. | |
| mPendingResponseQueue | Dictionary<int, PendingResponseQueueItem>: Pending queue per response. | |
| **Methods** | | |
| public WebSocketBaseTokenClient() | | Build a base token. |
| public WebSocketBaseTokenClient(WebSocketReliabilityOptions aReliabilityOptions) | | Build a base token whit reliability option. |
| public void SendTokenText(Token aToken) | | Send token text |
| public void SendTokenText(Token aToken, WebSocketResponseTokenListener aResponseListener) | | Send token text and wait for answer. |
| private void SendTokenBinary(Token aToken) | | Send token binary. |
| private void SendTokenBinary(Token aToken, WebSocketResponseTokenListener aResponseListener) | | Send token binary and wait for answer. |
| public void AddTokenClientListener(WebSocketClientTokenListener aTokenListener) | | Add listeners to the application. |
| public void RemoveTokenClientListener(WebSocketClientTokenListener aTokenListener) | | Remove listeners to the application. |

|  |  |  |
| --- | --- | --- |
| **Packet** | Token Client | |
| **Component** | WebSocketTokenClientListener | |
| **Class** | WebSocketClientTokenListener.cs | |
| **Extend** | WebSocketClientListener.cs | |
| **Description** | Listener to token client. | |
| **Dependencies** | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using ClientLibrary.org.jwebsocket.client.csharp.api; | |
| **Methods** | | |
| void ProcessOnTokenText(Token aToken) | | Process token text callback. |

|  |  |  |
| --- | --- | --- |
| **Packet** | Token Client | |
| **Component** | WebSocketResponseTokenListener | |
| **Class** | WebSocketResponseTokenListener.cs | |
| **Description** | Listener to response token. | |
| **Dependencies** | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text; | |
| **Methods** | | |
| void OnTimeout(Token aToken) | | Timeout callback |
| void OnResponse(Token aToken) | | Response callback |
| void OnSuccess(Token aToken) | | Success callback |
| void OnFailure(Token aToken) | | Failure callback |

|  |  |  |
| --- | --- | --- |
| **Packet** | Token Client | |
| **Component** | Token | |
| **Class** | DictionaryToken.cs | |
| **Extend** | Token.cs | |
| **Description** | Token implementation | |
| **Dependencies** | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using ClientLibrary.org.jwebsocket.client.token.api;  using ClientLibrary.org.jwebsocket.client.common; | |
| **Attributes** | | |
| mData | Dictionary<string, object>: Token data | |
| mBinary | Bool: True if the token is binary. | |
| **Methods** | | |
| public DictionaryToken() | | Build empty token. |
| public DictionaryToken(string aType) | | Build token whit token type. |
| public DictionaryToken(Dictionary<string, object> aDictionary) | | Build token whit data. |
| public DictionaryToken(string aNS, string aType) | | Build token whit name space and token type. |
| public void SetDictionary(Dictionary<string, object> aDictionary) | | Set token data. |
| public object GetObject(string aKey) | | Get object value for a key. |
| public string GetString(string aKey) | | Get string value for a key. |
| public void SetString(string aKey, string aValue) | | Set string value for a key. |
| public int GetInt(string aKey) | | Get int value for a key. |
| public void SetInt(string aKey, int aValue) | | Set int value for a key. |
| public double GetDouble(string aKey) | | Get double value for a key. |
| public void SetDouble(string aKey, double aValue) | | Set double value for a key. |
| public bool GetBool(string aKey) | | Get bool value for a key. |
| public void SetBool(string aKey, bool aValue) | | Set bool value for a key. |
| public List<object> GetList(string aKey) | | Get list value for a key. |
| public void SetList(string aKey, List<object> aList) | | Set list value for a key. |
| public void SetToken(string aKey, Token aToken) | | Set token value for a key. |
| public Token GetToken(string aKey) | | Get token value for a key. |
| public Dictionary<string, object> GetDictionary() | | Get token data. |
| public Dictionary<string, object> GetDictionary(string aKey) | | Get token data for a key. |
| public void SetDictionary(string aKey, Dictionary<string, object> aDictionary) | | Set token data for a key. |
| public string GetType() | | Get token type. |
| public void SetType(string aType) | | Set token type. |
| public string GetNS() | | Get name space. |
| public void SetNS(string aNS) | | Set name space. |
| public void Clear() | | Clear token data. |
| public void Remove(string aKey) | | Remove value for a key. |

|  |  |  |
| --- | --- | --- |
| **Packet** | Token Client | |
| **Component** | Tokenfactory | |
| **Class** | TokenFactory.cs | |
| **Description** | Build tokens. | |
| **Dependencies** | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using ClientLibrary.org.jwebsocket.client.token.api;  using ClientLibrary.org.jwebsocket.client.common;  using ClientLibrary.org.jwebsocket.client.csharp.api;  using ClientLibrary.org.jwebsocket.client.token.processor; | |
| **Methods** | | |
| public static Token CreateToken() | | Build empty token. |
| public static Token CreateToken(string aType) | | Build token whit token type. |
| public static Token CreateToken(string aNS, string aType) | | Build token whit name space and token type. |
| public static Token PacketToToken(string aFormat, WebSocketPacket aDataPacket) | | Converts a token packet. |
| public static WebSocketPacket TokenToPacket(string aFormat, Token aToken) | | Converts a token in a package. |

**Reusability**

The solution developed is structured primarily in 2 layers well defined and independent. One is the low level layer, which manages the communication between client and server using the protocol WebSocket. This layer is completely reusable for any other application you want to connect through the WebSocket protocol.

**Observer Pattern**

The observer pattern defines dependencies between objects one to many, so when an object changes state, all listeners are notified automatically. This pattern resolves the need to maintain consistency between related objects without requiring that classes are highly coupled.

By using the Observer pattern implementations are obtained easily readable, high flexibility and low coupling between classes in an OOP based system.

Implementing the Observer pattern in C # Client Library is regulated by the following interfaces:

**Interface for objects WebSocketClient listenable (subjects):**

public interface WebSocketClient

{

void Open(string aURL);

void Open(string aURL, string aSubProtocol);

void Open(string aURL, string aSubProtocol, int aTimeout);

void SendText(string aUTF8String);

void SendText(string aUTF8String, int aFragmentSize);

void SendBinary(byte[] aBinaryData);

void SendBinary(byte[] aBinaryData, int aFragmentSize);

void Ping();

void OnTextMessage(WebSocketPacket aDataPacket);

void OnBinaryMessage(WebSocketPacket aDataPacket);

void OnFragment(WebSocketPacket aFragment, int aIndex, int aTotal);

void OnOpen(WebSocketHeaders aHeader);

void OnClose(WebSocketCloseReason aCloseReason);

void OnError(WebSocketError aError);

void OnPing();

void OnPong();

void AddListener(WebSocketClientListener aListener);

void RemoveListener(WebSocketClientListener aListener);

}

**WebSocketClientListener interface for objects that want to listen to events from other objects (listeners)**

public interface WebSocketClientListener

{

void ProcessOnTextMessage(WebSocketPacket aDataPacket);

void ProcessOnBinaryMessage(WebSocketPacket aDataPacket);

void ProcessOnFragment(WebSocketPacket aFragment, int aIndex, int aTotal);

void ProcessOnOpen(WebSocketHeaders aHeader);

void ProcessOnClose(WebSocketCloseReason aCloseReason);

void ProcessOnError(WebSocketError aError);

void ProcessOnPing();

void ProcessOnPong();

}

**WebSocketClientTokenListener interface for objects that wish to listen to events from other objects (listeners)**

public interface WebSocketClientTokenListener : WebSocketClientListener

{

void ProcessOnTokenText(Token aToken);

}

**WebSocketResponseTokenListener interface for objects that want to listen to events from other objects (listeners)**

public interface WebSocketResponseTokenListener

{

void OnTimeout(Token aToken);

void OnResponse(Token aToken);

void OnSuccess(Token aToken);

void OnFailure(Token aToken);

}

**Example of using the Observer pattern in C # Client Library:**

|  |
| --- |
| class Program  {  static void Main(string[] args)  {  WebSocketBaseTokenClient mClient = new WebSocketBaseTokenClient(  new WebSocketReliabilityOptions(true, 3000, 3000) );  mClient.AddListener(new MyListener(mClient));  mClient.Open("ws://localhost:8787//jWebSocket//jWebSocket");  }  }  public class MyListener : WebSocketClientTokenListener  {  private WebSocketBaseTokenClient mClient;  public MyListener(WebSocketBaseTokenClient aClient)  {  this.mClient = aClient;  }  public void ProcessOnTokenText(Token aToken)  {  Console.WriteLine("Token Text: " + aToken.GetString("data"));  }  public void ProcessOnBinaryMessage(WebSocketPacket aDataPacket) { }  public void ProcessOnClose(WebSocketCloseReason aCloseReason)  {  Console.WriteLine(aCloseReason.ToString());  }  public void ProcessOnError(WebSocketError aError)  {  Console.WriteLine(aError.Reason);  }  public void ProcessOnFragment(WebSocketPacket aFragment, int aIndex,  int aTotal) { }  public void ProcessOnOpen(WebSocketHeaders aHeader)  {  Token lMyToken = TokenFactory.CreateToken(  WebSocketMessage.NS\_SYSTEM\_PLUGIN, WebSocketMessage.ECHO);  lMyToken.SetString("data", "tito");  mClient.SendTokenText(lMyToken, new MyResponse());  }  public void ProcessOnPing() { }  public void ProcessOnPong() { }  public void ProcessOnTextMessage(WebSocketPacket aDataPacket) { }  }  public class MyResponse : WebSocketResponseTokenListener  {  public void OnFailure(Token aToken)  {  Console.WriteLine("Failure");  }  public void OnResponse(Token aToken) { }  public void OnSuccess(Token aToken)  {  Console.WriteLine("Success");  }  public void OnTimeout(Token aToken) { }  } |

## Interfaces

The solution does not contain interfaces.

## Frameworks, libraries and tools

**Framework**:

. NET Framework 4.0: Contains all the necessary libraries to develop and run applications based on this technology.

**Libraries:**

JSON under the GPL: This program is used to serialize and deserialize data exchanged with the server through the WebSocket protocol.

Log4net under the Apache License: This library is used to display all the logs of the developed solution.

## Database and data persistence

This solution does not contain data persistence.

## Hardware

You do not need any special hardware to develop this solution.

## Security

## It uses SSL for secure transfer.

## Testing, quality assurance and continuous integration

No testing

## Continuous improvement

There are proposed methods and strategies for continuous improvement of the solution.

## Reference

|  |
| --- |
| <?xml version="1.0"?>  <doc>  <assembly>  <name>ClientLibrary</name>  </assembly>  <members>  <member name="T:ClientLibrary.org.jwebsocket.client.token.kit.WebSocketTokenListener">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.token.api.WebSocketClientTokenListener">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Interface for the low level WebSocket listeners.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener.ProcessOnTextMessage(ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket)">  <summary>  Invoked when a data packet as text message from a client is received.  </summary>  <param name="aDataPacket">Data packet received.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener.ProcessOnBinaryMessage(ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket)">  <summary>  Invoked when a data packet as binary message from a client is received.  </summary>  <param name="aDataPacket">Data packet received.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener.ProcessOnFragment(ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket,System.Int32,System.Int32)">  <summary>  Invoked when a fragment from client is received.  </summary>  <param name="aFragment">Data packet fragment received.</param>  <param name="aIndex">Index of fragment.</param>  <param name="aTotal">Total size of fragment.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener.ProcessOnOpen(ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders)">  <summary>  Invoked when a new client connects to the Client.  </summary>  <param name="aHeader">Header fields from the handshake.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener.ProcessOnClose(ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketCloseReason)">  <summary>  Invoked when a client was disconnted to the Client.  </summary>  <param name="aCloseReason">Represent the reason of the disconnect.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener.ProcessOnError(ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketError)">  <summary>  Invoked when any error occurs.  </summary>  <param name="aError">Description of the error.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener.ProcessOnPing">  <summary>  Invoked when client sent a ping.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener.ProcessOnPong">  <summary>  Invoked when server sent a pong.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  API for low level client  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.Open(System.String)">  <summary>  Establish a connection to a websocket server.  </summary>  <param name="aURL">Server URL.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.Open(System.String,System.String)">  <summary>  Establish a connection to a websocket server.  </summary>  <param name="aURL">Server URL.</param>  <param name="aSubProtocol">WebSocket protocol specification.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.Open(System.String,System.String,System.Int32)">  <summary>  Establish a connection to a websocket server.  </summary>  <param name="aURL">Server URL.</param>  <param name="aSubProtocol">WebSocket protocol specification.</param>  <param name="aTimeout">Timeout for close the conection.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.SendText(System.String)">  <summary>  Send a complete packet as UTF8 string.  </summary>  <param name="aUTF8String">Packet as UTF8 String.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.SendText(System.String,System.Int32)">  <summary>  Send packet UTF8 string as multiple fragments.  </summary>  <param name="aUTF8String">Packet as UTF8 String.</param>  <param name="aFragmentSize">Maximum fragment size.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.SendBinary(System.Byte[])">  <summary>  Send a complete packet as binary data.  </summary>  <param name="aBinaryData">Binary data.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.SendBinary(System.Byte[],System.Int32)">  <summary>  Send packet binary Data as multiple fragments.  </summary>  <param name="aBinaryData">Binary data.</param>  <param name="aFragmentSize">Maximum fragment size.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.Ping">  <summary>  Send a ping frame to the server and starts a timeout observer.  </summary>  <param name="aTimeout">Timeout for close the conection.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.OnTextMessage(ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket)">  <summary>  Called when a text message has been received.  </summary>  <param name="aDataPacket">Data packet received .</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.OnBinaryMessage(ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket)">  <summary>  Called when a binary message has been received.  </summary>  <param name="aDataPacket">Data packet received.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.OnFragment(ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket,System.Int32,System.Int32)">  <summary>  Called when a fragment has been received.  </summary>  <param name="aFragment">Data packet fragment received.</param>  <param name="aIndex">Index of fragment.</param>  <param name="aTotal">Total size of fragment.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.OnOpen(ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders)">  <summary>  Called when connection to a websocket server has been Establish.  </summary>  <param name="aHeader">Header fields from the handshake.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.OnClose(ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketCloseReason)">  <summary>  Called when connection to a websocket server has been closed.  </summary>  <param name="aCloseReason">Represent the reason of the disconnect.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.OnError(ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketError)">  <summary>  Called when occurs any error.  </summary>  <param name="aError">Represents the error occurred.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.OnPing">  <summary>  Called when client sent a ping.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.OnPong">  <summary>  Called when server sent a pong.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.AddListener(ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener)">  <summary>  Adds the listener.  </summary>  <param name="aListener">A listener.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.RemoveListener(ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener)">  <summary>  Removes the listener.  </summary>  <param name="aListener">A listener.</param>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketStatus">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  WebSocket Status.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.common.WebSocketTypeEncoding">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Type encoding  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.common.WebSocketConvert">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Convert strings to bytes and viceversa.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.common.WebSocketConvert.StringToBytes(System.String,ClientLibrary.org.jwebsocket.client.common.WebSocketTypeEncoding)">  <summary>  Strings to bytes.  </summary>  <param name="aString">String.</param>  <param name="aEncoding">Encoding.</param>  <returns>Array bytes</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.common.WebSocketConvert.BytesToString(System.Byte[],ClientLibrary.org.jwebsocket.client.common.WebSocketTypeEncoding)">  <summary>  Bytes to strings.  </summary>  <param name="aBytes">Bytes.</param>  <param name="aEncoding">Encoding.</param>  <returns>String</returns>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.token.tbase.DictionaryToken">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.token.api.Token">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetDictionary(System.Collections.Generic.Dictionary{System.String,System.Object})">  <summary>  Copies all fields from a Map into the Token. A check has to be made  by the corresponding implementations that only such data types are  passed that are supported by the Token abstraction.  </summary>  <param name="aDictionary">Dictionary.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetObject(System.String)">  <summary>  Gets the object.  </summary>  <param name="aKey">key.</param>  <returns></returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetString(System.String)">  <summary>  Gets the string.  </summary>  <param name="aKey">key.</param>  <returns></returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetString(System.String,System.String)">  <summary>  Sets the string.  </summary>  <param name="aKey">key.</param>  <param name="aValue">Value.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetInt(System.String)">  <summary>  Gets the int.  </summary>  <param name="aKey">key.</param>  <returns></returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetInt(System.String,System.Int32)">  <summary>  Sets the int.  </summary>  <param name="aKey">key.</param>  <param name="aValue">Value.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetDouble(System.String)">  <summary>  Gets the double.  </summary>  <param name="aKey">key.</param>  <returns></returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetDouble(System.String,System.Double)">  <summary>  Sets the double.  </summary>  <param name="aKey">key.</param>  <param name="aValue">Value.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetBool(System.String)">  <summary>  Gets the bool.  </summary>  <param name="aKey">key.</param>  <returns></returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetBool(System.String,System.Boolean)">  <summary>  Sets the bool.  </summary>  <param name="aKey">key.</param>  <param name="aValue">if set to <c>true</c> [a value].</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetList(System.String)">  <summary>  Gets the list.  </summary>  <param name="aKey">key.</param>  <returns>List.</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetList(System.String,System.Collections.Generic.List{System.Object})">  <summary>  Sets the list.  </summary>  <param name="aKey">key.</param>  <param name="aList">List.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetToken(System.String,ClientLibrary.org.jwebsocket.client.token.api.Token)">  <summary>  Sets the token.  </summary>  <param name="aKey">key.</param>  <param name="aToken">Token.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetToken(System.String)">  <summary>  Gets the token.  </summary>  <param name="aKey">key.</param>  <returns></returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetDictionary">  <summary>  Gets the dictionary.  </summary>  <returns></returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetDictionary(System.String)">  <summary>  Gets the dictionary.  </summary>  <param name="aKey">key.</param>  <returns>Dictionary.</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetDictionary(System.String,System.Collections.Generic.Dictionary{System.String,System.Object})">  <summary>  Sets the dictionary.  </summary>  <param name="aKey">key.</param>  <param name="aDictionary">Dictionary.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetType">  <summary>  Gets the type.  </summary>  <returns>Type.</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetType(System.String)">  <summary>  Sets the type.  </summary>  <param name="aType">Type.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetNS">  <summary>  Gets the NS.  </summary>  <returns>NS.</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetNS(System.String)">  <summary>  Sets the NS.  </summary>  <param name="aNS">NS.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.Clear">  <summary>  Resets all fields of the token. After this operation the token is empty.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.Remove(System.String)">  <summary>  Removes the specified a key.  </summary>  <param name="aKey">key.</param>  </member>  <member name="P:ClientLibrary.org.jwebsocket.client.token.api.Token.IsBinary">  <summary>  Determines whether this instance is binary.  </summary>  <returns><c>true</c> if this instance is binary; otherwise, <c>false</c>.</returns>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketFrameType">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Defines an frame type.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketCloseReason">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  1000 indicates a normal closure, meaning that the purpose for  which the connection was established has been fulfilled.    1001 indicates that an endpoint is "going away", such as a server  going down or a browser having navigated away from a page.    1002 indicates that an endpoint is terminating the connection due  to a protocol error.    1004 Reserved. The specific meaning might be defined in the future.    1005 is a reserved value and MUST NOT be set as a status code in a  Close control frame by an endpoint. It is designated for use in  applications expecting a status code to indicate that no status  code was actually present.    1006 is a reserved value and MUST NOT be set as a status code in a  Close control frame by an endpoint. It is designated for use in  applications expecting a status code to indicate that the  connection was closed abnormally, e.g., without sending or  receiving a Close control frame.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.csbase.WebSocketBaseClient">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Base WebSocket implementation.  This uses thread model for handling WebSocket connection which is defined  by the WebSocket protocol specification.  http://www.whatwg.org/specs/web-socket-protocol/  http://www.w3.org/TR/websockets/  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.common.WebSocketMessage">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  All messages of the client.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Specifies the API for low level data packets which are interchanged between  client and server. Data packets do not have a special format at this  communication level.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.InitFragmented(System.Int32)">  <summary>  Inits the fragmented.  </summary>  <param name="aTotal">Total size.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.SetFragment(System.String,System.Int32)">  <summary>  Sets the fragment.  </summary>  <param name="aString">String fragment.</param>  <param name="aIdx">Idx.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.PackFragments">  <summary>  Packs the fragments.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.SetString(System.String)">  <summary>  Sets the value of the data packet to the given string by using  default encoding.  </summary>  <param name="aString">String value.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.SetString(System.String,ClientLibrary.org.jwebsocket.client.common.WebSocketTypeEncoding)">  <summary>  Sets the value of the data packet to the given string by using  the passed encoding.  </summary>  <param name="aString">String value.</param>  <param name="aEncoding">Encoding type.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.SetUTF8(System.String)">  <summary>  Sets the value of the data packet to the given string by using  UTF-8 encoding.  </summary>  <param name="aString">String value.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.SetASCII(System.String)">  <summary>  Sets the value of the data packet to the given string by using  7 bit US-ASCII encoding.  </summary>  <param name="aString">String value.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.GetString">  <summary>  Returns the content of the data packet as a string using default  encoding.  </summary>  <returns>Raw Data packet as string with default encoding</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.GetString(ClientLibrary.org.jwebsocket.client.common.WebSocketTypeEncoding)">  <summary>  Returns the content of the data packet as a string using the passed  encoding.  </summary>  <param name="aEncoding">Encoding type.</param>  <returns>Raw Data packet as string using passed encoding</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.GetUTF8">  <summary>  Interprets the data packet as a UTF8 string and returns the string  in UTF-8 encoding.If an exception occurs "null" is returned.  </summary>  <returns>Data packet as UTF-8 string or <c>null</c> if not convertible.</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.GetASCII">  <summary>  Interprets the data packet as a US-ASCII string and returns the string  in US-ASCII encoding. If an exception occurs "null" is returned.  </summary>  <returns>Data packet as US-ASCII string or <c>null</c> if not convertible.</returns>  </member>  <member name="P:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.FrameType">  <summary>  Gets or sets data packet.  </summary>  <value>The type of the frame.</value>  </member>  <member name="P:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.ByteArray">  <summary>  Gets or sets the byte array.  </summary>  <value>The byte array.</value>  </member>  <member name="P:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.Fragments">  <summary>  Gets or sets the fragments.  </summary>  <value>The fragments.</value>  </member>  <member name="P:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.IsFragmented">  <summary>  Gets or sets a value indicating whether this instance is fragmented.  </summary>  <value><c>true</c> if this instance is fragmented; otherwise, <c>false</c>.</value>  </member>  <member name="P:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.IsComplete">  <summary>  Gets or sets a value indicating whether this instance is complete.  </summary>  <value><c>true</c> if this instance is complete; otherwise, <c>false</c>.</value>  </member>  <member name="P:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.CreationDate">  <summary>  Gets or sets the creation date.  </summary>  <value>The creation date.</value>  </member>  <member name="P:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.Timeout">  <summary>  Gets or sets the timeout.  </summary>  <value>The timeout.</value>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketExceptionType">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Defines an type of exception.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.token.kit.PendingResponseQueueItem">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketRuntimeException">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Define Runtime Exception for Websocket.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketTimeout">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketTimeout.CallWithTimeout(System.Action{System.Net.Sockets.NetworkStream},System.Int32,System.Net.Sockets.NetworkStream)">  <summary>  Calls the with timeout.  </summary>  <param name="aAction">Action.</param>  <param name="aTimeoutMilliseconds">Timeout milliseconds.</param>  <param name="aIn">Network Stream.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketTimeout.CallWithTimeout(System.Action{ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket},System.Int32,ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket)">  <summary>  Calls the with timeout.  </summary>  <param name="aAction">Action.</param>  <param name="aTimeoutMilliseconds">Timeout milliseconds.</param>  <param name="lPacket">WebSocket Packet.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketTimeout.CallWithTimeout(System.Action{ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders},System.Int32,ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders)">  <summary>  Calls the with timeout.  </summary>  <param name="aAction">Action.</param>  <param name="aTimeoutMilliseconds">Timeout milliseconds.</param>  <param name="lHeaders"> WebSocket Headers.</param>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketError">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Defines an error  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketProtocolAbstraction">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Utility class for packetizing WebSocketPacket into web socket protocol packet or packets (with fragmentation) and  vice versa.    Web socket protocol packet specification  http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-03)    0 1 2 3  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1  +-+-+-+-+-------+-+-------------+-------------------------------------------+  |M|R|R|R| opcode|R| Payload len | Extended payload length |  |O |S|S|S | (4) |S| (7) | (16/63) |  |R |V|V|V | |V| | (if payload len==126/127) |  |E |1 |2|3 | |4| | |  +-+-+-+-+-------+-+-------------+ - - - - - - - - ---------------- - - - - - - +  | Extended payload length continued, if payload len == 127 |  + - - - - - - - - - - - - - - - +--------------------------------------------------+  | | Extension data |  +-------------------------------+ - - - - - - - - - - --------------------- - - - - +  : :  +---------------------------------------------------------------------------------+  : Application data :  +---------------------------------------------------------------------------------+  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketException">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Defines an exception  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketStateOfStatus">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Status of the connection.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.token.api.WebSocketResponseTokenListener">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.WebSocketResponseTokenListener.OnTimeout(ClientLibrary.org.jwebsocket.client.token.api.Token)">  <summary>  Is fired when the given response timeout is exceeded.  </summary>  <param name="aToken"></param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.WebSocketResponseTokenListener.OnResponse(ClientLibrary.org.jwebsocket.client.token.api.Token)">  <summary>  Is fired on any response to a send token.  </summary>  <param name="aToken"></param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.WebSocketResponseTokenListener.OnSuccess(ClientLibrary.org.jwebsocket.client.token.api.Token)">  <summary>  Is fired if token.code equals 0 (zero).  </summary>  <param name="aToken"></param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.WebSocketResponseTokenListener.OnFailure(ClientLibrary.org.jwebsocket.client.token.api.Token)">  <summary>  Is fired if token.code does not equal 0 (zero).  </summary>  <param name="aToken"></param>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHandshake">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Utility class for all the handshake.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHandshake.GenerateC2SRequest">  <summary>  Generates the initial Handshake from a Client to the WebSocket.  </summary>  <returns>Handshake as byte array.</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHandshake.VerifyS2CResponse(ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders)">  <summary>  Verify that the server's response is correct.  </summary>  <param name="aHeaders">Header with the response data.</param>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.token.processor.JSONTokenProcessor">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.csbase.WebSocketRawPacket">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Implements the low level data packets which are interchanged between  client and server. Data packets do not have a special format at this  communication level.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.token.tbase.WebSocketBaseTokenClient">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Token based implementation of <c>WebSocketBaseClient</c>.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketSubProtocol">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Define WebSocket sub protocol.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Contains all constants of the client.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.mLog">  <summary>  Logger.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_ENCODING\_DEFAULT">  <summary>  Use text format as default encoding for WebSocket Packets if not explicitly specified.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_VERSION\_DEFAULT">  <summary>  WebSocket default protocol version.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_SUBPROT\_PREFIX">  <summary>  jWebSocket sub protocol prefix.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_SUBPROT\_JSON">  <summary>  jWebSocket JSON sub protocol.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_SUBPROT\_CSV">  <summary>  jWebSocket CSV sub protocol.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_SUBPROT\_XML">  <summary>  jWebSocket XML sub protocol.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_SUBPROT\_TEXT">  <summary>  jWebSocket custom specific text sub protocol.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_SUBPROT\_BINARY">  <summary>  jWebSocket custom specific binary sub protocol.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_FORMAT\_JSON">  <summary>  JSON sub protocol format.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_FORMAT\_CSV">  <summary>  CSV sub protocol format.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_FORMAT\_XML">  <summary>  XML sub protocol format.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_FORMAT\_BINARY">  <summary>  Binary sub protocol format.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_FORMAT\_TEXT">  <summary>  Custom specific sub protocol format.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_SUBPROT\_DEFAULT">  <summary>  Default protocol.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.HOST">  <summary>  Host header from handshake.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.UPGRADE">  <summary>  Upgrade header from handshake.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.CONNECTION">  <summary>  Connection header from handshake.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.SEC\_WEBSOCKET\_KEY">  <summary>  Sec-WebSocket-Key header from handshake.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.ORIGIN">  <summary>  Sec-WebSocket-Origin header from handshake.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.SEC\_WEBSOCKET\_PROTOCOL">  <summary>  Sec-WebSocket-Protocol header from handshake.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.SEC\_WEBSOCKET\_VERSION">  <summary>  Sec-WebSocket-Version header from handshake.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.SEC\_WEBSOCKET\_ACCEPT">  <summary>  Sec-WebSocket-Accept header from handshake.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.SCHEME\_WS">  <summary>  Scheme ws for connection.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.SCHEME\_WSS">  <summary>  Scheme wss for connection.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.CONFIG">  <summary>  XML configuration file.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.DEFAULT\_MAX\_FRAME\_SIZE">  <summary>  The default maximum frame size if not configured.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.DEFAULT\_OPEN\_TIMEOUT">  <summary>  Default Session Timeout for client connections.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.DEFAULT\_PING\_TIMEOUT">  <summary>  Default ping Timeout for client connections.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.DEFAULT\_PING\_DELAY">  <summary>  Default ping delay for client connections.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_SUPPORTED\_HYBI\_VERSIONS">  <summary>  WebSocket supported hixie versions.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.#cctor">  <summary>  Initializes the <see cref="T:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants"/> class.  </summary>`  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.InitialiceXML">  <summary>  Initialices the XML config.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.token.tbase.TokenFactory">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketEncoding">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Type of encoding for data packet.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketCookieManager">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>4/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.api.ICookiesManager">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>4/26/2012</lastUpdate>  <summary>  API for Cookie Manage  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.ICookiesManager.AddCookies(System.Collections.Generic.List{System.String},System.Uri)">  <summary>    </summary>  <param name="aCookies"></param>  <param name="aUri"></param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.ICookiesManager.GetCookies(System.Uri)">  <summary>    </summary>  <param name="aUri"></param>  <returns></returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.ICookiesManager.ProcessCookies(System.Web.HttpCookieCollection)">  <summary>    </summary>  <param name="aCookies"></param>  <returns></returns>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketReliabilityOptions">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Reliability Options for connection.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Implementation of the request and response headers  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders.ReadResponseFromStream(System.Net.Sockets.NetworkStream)">  <summary>  Reads the response from stream.  </summary>  <param name="aSR">Data stream from server.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders.ReadRequestFromBuffer(System.Byte[])">  <summary>  Reads the request from buffer.  </summary>  <param name="aBuff">Data buffer from client.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders.ToStringRequest">  <summary>  Gets request as string.  </summary>  <returns>String request</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders.ToStringResponse">  <summary>  Gets response as string.  </summary>  <returns>String response</returns>  </member>  </members>  </doc> |