WebSocket Documentation

Andrea Vaccaro - David Grollier

Contents

[1 Description 2](#_Toc146629938)

[2 Library 2](#_Toc146629939)

[2.1 Actor 2](#_Toc146629940)

[2.1.1 WebSocket Client.lvclass 2](#_Toc146629941)

[2.1.2 WebSocket Connection.lvclass 2](#_Toc146629942)

[2.1.3 WebSocket Server Manager.lvclass 3](#_Toc146629943)

[2.1.4 WebSocket Server.lvclass 3](#_Toc146629944)

[2.1.5 WebSocket Service.lvclass 3](#_Toc146629945)

[2.2 Abstract message 4](#_Toc146629946)

[2.2.1 WebSocket Connection Handler Msg.lvclass 4](#_Toc146629947)

[2.2.2 WebSocket Server-Service Enqueuer Msg.lvclass 4](#_Toc146629948)

[3 Server implementation 4](#_Toc146629949)

[3.1 API 5](#_Toc146629950)

[3.1.1 WebSocket Server Listener Settings 5](#_Toc146629951)

[3.1.2 With the WebSocket Server Manager 5](#_Toc146629952)

[3.1.3 Without the WebSocket Server Manager 6](#_Toc146629953)

[3.1.4 Add Service 7](#_Toc146629954)

[3.1.5 Broadcast 8](#_Toc146629955)

[4 Client implementation 9](#_Toc146629956)

[4.1 API 9](#_Toc146629957)

# Description

Library implementing a high-level actor-based implementation of WebSocket communication. An actor can subscribe to the WebSocket Actor to facilitate WebSocket-based communication. It can serve both as a client and a server.

# Library

A screenshot of a computer

Description automatically generated

## Actor

### WebSocket Client.lvclass

Manages the definition of WebSocket services on a specified server. Services can be defined by the subscriber by sending the "Add Service" message to the actor. The subscriber must provide a data handler message in form of a concrete implementation of the "WebSocket Connection Handler Msg" abstract message used to handle the connection events corresponding to the WebSocket communication and a concrete implementation of the "WebSocket Server-Service Enqueuer Msg" abstract message used to receive the WebSocket service actor enqueuer. For each service defined a "WebSocket Service" actor is started that will send data to the subscriber that has defined the service. The server is characterized by an interface and a port on which it listens. The actor listens according to the server specifications on a port and on an interface if not all, when a connection is established, the actor performs the WebSocket handshake. If the service requested is defined in this server, we inform the WebSocket service Actor who will start a corresponding WebSocket connection actor. If either the service is not defined, or the handshake fails, we do nothing, the WebSocket protocol will take care of communicating the failure to the client.

### WebSocket Connection.lvclass

### WebSocket Server Manager.lvclass

This actor's role is solely that of launching and managing the stop of "WebSocket Server" actors listening on at a specific port and interface address ("Launch Server Msg" and "Stop Server Msg" messages). The actor will ensure that launched "WebSocket Server" doesn’t conflict on interface and port resources. If the server is already running at that port or on the specified interface the method sends just the server enqueuer to the client. If the server is not running it will be launched.

### WebSocket Server.lvclass

Manages the definition of WebSocket services on a specified server. Services can be defined by the subscriber by sending the "Add Service" message to the actor. The subscriber must provide a data handler message in form of a concrete implementation of the "WebSocket Connection Handler Msg" abstract message used to handle the connection events corresponding to the WebSocket communication and a concrete implementation of the "WebSocket Server-Service Enqueuer Msg" abstract message used to receive the WebSocket service actor enqueuer. For each service defined a "WebSocket Service" actor is started that will send data to the subscriber that has defined the service. The server is characterized by an interface and a port on which it listens. The actor listens according to the server specifications on a port and on an interface if not all, when a connection is established, the actor performs the WebSocket handshake. If the service requested is defined in this server, we inform the WebSocket service Actor who will start a corresponding WebSocket connection actor. If either the service is not defined, or the handshake fails, we do nothing, the WebSocket protocol will take care of communicating the failure to the client.

### WebSocket Service.lvclass

It sends packets received by the connected WebSocket Services to the WebSocket client subscriber as "WebSocket Connection Handler Msg" messages. It allows the subscriber to send data to a specified client ("Send to Peer Msg" message of the parent "WebSocket Connection" class). Upon starting the actor, the subscriber must provide a data handler message in form of a concrete implementation of the "WebSocket Connection Handler Msg" abstract message used to handle the connection events corresponding to the WebSocket communication and a concrete implementation of the "WebSocket Server-Service Enqueuer Msg" abstract message used to receive the WebSocket service actor enqueuer.

## Abstract message

### WebSocket Connection Handler Msg.lvclass

This abstract message is sent by either the WebSocket Client or by the WebSocket Service to the caller Actor on four events:

* Connect
* Disconnect
* Message
* Drop

For each caller Actor a concrete subclass of the message should be created by overriding the methods:

* OnConnect
* OnDisconnect
* OnMessage
* OnDrop

The caller actor can program the required action for the corresponding event.

### WebSocket Server-Service Enqueuer Msg.lvclass

This abstract message is used either by the WebSocket Server Manager or by the WebSocket Server to send the just launched/added Server/Service Enqueuer to the actor requesting the operation. Each actor requesting the operation should subclass this message and then override the Do.vi in which the Server/Service Enqueuer is stored in the requesting actor.

# Server implementation

The WebSocket Server can handle multiple services (also called resource names or endpoint), allowing it to broadcast messages to all clients connected to the same service. The actor subscriber can also implement actions on specific events of the WebSocket Client.

## API

### WebSocket Server Listener Settings

A screenshot of a computer

Description automatically generated

**timeout ms** specifies the time, in milliseconds, that the function waits for a connection. If a connection is not established in the specified time, the function returns an error. The default value is –1, which indicates to wait indefinitely.

**resolve remote address** indicates whether to call the IP To String function on the remote address.

**net address** specifies on which network address to listen. Specifying an address is useful if you have more than one network card, such as two Ethernet cards, and want to listen only on the card with the specified address. If you do not specify a network address, LabVIEW listens on all network addresses.

**port** is the port number on which you want to listen for a connection.

### With the WebSocket Server Manager

A screenshot of a computer

Description automatically generated

**Subscriber Enqueuer** specifies the enqueuer of the actor (must be created) that subscribes to the server actor.

**Message Priority** specifies the placement of the Message in the message queue. This placement determines when the actor will process the message.

|  |  |
| --- | --- |
| 0 | Low—Specifies that the message will be processed after messages of all other priorities. Multiple low-priority messages are processed in the order they are sent. |
| 1 | Normal (Default)—Specifies that the message will be processed after critical- and high-priority messages but before low-priority ones. Multiple normal-priority messages are processed in the order they are sent. |
| 2 | High—Specifies that the message will be processed first. Multiple high-priority messages are processed in the order they are sent. High-priority messages can be superseded only by an Emergency Stop or Last Ack message, both of which have critical priority. |

**Message Enqueuer** specifies the enqueuer of the WebSocket Server Manager actor.

**Listener Settings** specifies the settings for the Server.

**Handshake timeout** specifies the timeout of the handshake when the client establishes a WebSocket connection to the server.

**Receive Server Enqueuer Handler** specifies the abstract message (must be created) child of the abstract message “*WebSocket Server-Service Enqueuer Msg.lvclass*”.

### Without the WebSocket Server Manager

The following settings of the WebSocket Server actor can be updated (the port must be entered)

A close-up of a sign

Description automatically generated

**Listener Settings** specifies the settings for the Server.

**Handshake timeout** specifies the timeout of the handshake when the client establishes a WebSocket connection to the server.

### Add Service

A screenshot of a computer

Description automatically generated

**Receive Connection Enqueuer** specifies the abstract message (must be created) child of the abstract message “*WebSocket Server-Service Enqueuer Msg.lvclass”.*

**Message Priority** specifies the placement of the Message in the message queue. This placement determines when the actor will process the message.

|  |  |
| --- | --- |
| 0 | Low—Specifies that the message will be processed after messages of all other priorities. Multiple low-priority messages are processed in the order they are sent. |
| 1 | Normal (Default)—Specifies that the message will be processed after critical- and high-priority messages but before low-priority ones. Multiple normal-priority messages are processed in the order they are sent. |
| 2 | High—Specifies that the message will be processed first. Multiple high-priority messages are processed in the order they are sent. High-priority messages can be superseded only by an Emergency Stop or Last Ack message, both of which have critical priority. |

**Message Enqueuer** specifies the enqueuer of the WebSocket Server actor.

**Service Name** specifies the name of the service.

**Subscriber Enqueuer** specifies the enqueuer of the actor (must be created) that subscribes to the server actor.

**WebSocket Data Handler** specifies the abstract message (must be created) child of the abstract message “*WebSocket Connection Handler Msg.lvclass”.*

### Broadcast

A screenshot of a computer

Description automatically generated  
**Message Priority** specifies the placement of the Message in the message queue. This placement determines when the actor will process the message.

|  |  |
| --- | --- |
| 0 | Low—Specifies that the message will be processed after messages of all other priorities. Multiple low-priority messages are processed in the order they are sent. |
| 1 | Normal (Default)—Specifies that the message will be processed after critical- and high-priority messages but before low-priority ones. Multiple normal-priority messages are processed in the order they are sent. |
| 2 | High—Specifies that the message will be processed first. Multiple high-priority messages are processed in the order they are sent. High-priority messages can be superseded only by an Emergency Stop or Last Ack message, both of which have critical priority. |

**Message Enqueuer** specifies the enqueuer of the WebSocket Service actor.

**Data** specifies the data to be sent.

# Client implementation

The WebSocket Client can send messages to the other client connected to the same service. The actor subscriber can also implement actions on specific events of the WebSocket Client.

## API

The following settings of the WebSocket Client actor must be updated:

A screenshot of a computer

Description automatically generated

**WS URI** specifies the URI of the client with the following format:

*ws://hostname:port/service*

**Timeout (ms)** is the timeout to establish the connection to the server. A value of –1 indicates to wait indefinitely.

**WebSocket Data Handler** specifies the abstract message (must be created) child of the abstract message “*WebSocket Connection Handler Msg.lvclass”.*

**Subscriber Enqueuer** specifies the enqueuer of the actor (must be created) that subscribes to the client actor.