TMS FNC WebSocket

Developer guide

tmssoftware.com

Table of contents

1. Getting started

1.1 Overview

TMS FNC WebSocket is cross-platform and cross-framework component library for web socket communication.

1.1.1 Availability

Supported frameworks and platforms

- VCL: Win32/Win64
- FMX: Win32/Win64, macOS, iOS, Linux, Android
- WEB: Chrome, Edge, Firefox, Safari…

Supported IDE's

- Delphi XE7 and C++ Builder XE7 or newer releases

1.2 Installation

1.2.1 Automatic installation (FMX and VCL only)

Unzip the distribution file `TMSFNCWebSocketSetup.zip` into a separate directory, further referred to as `{$TMSDIR}`

Run the setup executable file and install the components for the correct environment. If the setup for FMX or VCL fails follow the manual installation instructions below.

1.2.2 Manual installation

FMXLinux (tested in Ubuntu 20.04)

FMXLinux is not installed out of the box in RAD Studio. Please follow the instructions here to install FMXLinux.

After following the above instructions, please execute the following commands

```
sudo apt install joe wget p7zip-full curl openssh-server build-essential zlib1g-dev libcurl4-gnutls-dev
libncurses5
sudo apt-get install zlib1g-dev
sudo apt install libgl1-mesa-glx libglu1-mesa libgtk-3-common libgstreamer1.0-0 libgstreamer-plugins-
base1.0-0
sudo apt install libwebkit2gtk-4.0-dev
```

For Delphi XE7, XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11

In the IDE, select File, Open and browse for `FMXTMSFNCWebSocketPkgD*.dproj` From the project manager, right click on the `FMXTMSFNCWebSocketPkgD*.dproj` project and from the menu, choose install

In the IDE, select File, Open and browse for `VCLTMSFNCWebSocketPkgD*.dproj` From the project manager, right click on the `VCLTMSFNCWebSocketPkgD*.dproj` project and from the menu, choose install

In the IDE, select File, Open and browse for `TMSFNCWebSocketPkgDED*.dproj` From the project manager, right click on the `TMSFNCWebSocketPkgDED*.dproj` project and from the menu, choose install

(* XE7, XE8, XE9, XE10, XE11, XE12, XE13 or XE14)

2. Reference

2.1 TTMSFNCWebSocketServer

The `TTMSFNCWebSocketServer` is a non-visual component enabling communication to websocket clients. It implements the RFC 6455 standard for websocket communication protocol.

2.1.1 Start/stop the server

Starting/stopping the server is as simple as setting the `Active` property.

```
procedure TForm1.startBtnClick(Sender: TObject);
begin
  TMSFNCWebSocketServer1.Active := True;
end;

procedure TForm1.stopBtnClick(Sender: TObject);
begin
  TMSFNCWebSocketServer1.Active := False;
end;
```

2.1.2 Allowing connections

By default all connections are allowed as long as they initiate a valid handshake request. After the handshake request is validated, the `OnAllow` event is triggered. This event has an `AAllow: Boolean` parameter which can be used to allow or disallow a connection regardless of the handshake request.

```
procedure TForm2.TMSFNCWebSocketServer1Allow(Sender: TObject;
  AConnection: TTMSFNCServerWebSocketConnection; var aAllow: Boolean);
begin
  //Allow all connections
  AAllow := True
end;
```

2.1.3 Send data

Use `BroadcastMessage(AMessage: string)` or `BroadcastData(AData: TBytes)` to broadcast a string or binary messages to all connected clients.

```
procedure TForm1.broadcastBtnClick(Sender: TObject);
begin
  TMSFNCWebSocketServer1.BroadcastMessage('Hello client!');
end;
```

Use `SendMessageTo(AMessage, ASelector)` or `SendDataTo(AData, ASelector)` to send string or binary messages to connected clients. The `ASelector` callback determines which clients should the message be sent to.

```
//Forward incoming message to all client, except the one the message came from
procedure TForm2.TMSFNCWebSocketServer1MessageReceived(Sender: TObject;
```

```
  AConnection: TTMSFNCWebSocketConnection; const aMessage: string);
begin
  TMSFNCWebSocketServer1.SendMessageTo(AMessage, function(AClientConnection:
TTMSFNCServerWebSocketConnection): Boolean
    begin
      Result := AClientConnection <> AConnection;
    end);
end;
```

Send in multiple frames

By default all messages are sent in one single frame. If `SplitMessage` is set to `True`, the message will be split up into multiple frames based on size defined by `FrameSize`. Size of the frame includes frame headers too.

2.1.4 Receive data/messages

When a string message arrives from one of the clients the `OnMessageReceived` event will be triggered. The `AMessage` parameter will contain the string message from the client.

When a binary message arrives from one of the clients the `OnBinaryDataReceived` event will be triggered. The `AData` parameter will contain the binary message from the client.

2.1.5 Ping and pong

Whenever a ping message arrives from a client the `OnPing` event is triggered and an automatic pong message is sent back.

If the `Options` property contains the `twsoManualPong` value the poing message is **NOT** sent automatically. In this case the `OnPing` can be used to send manually or skip it entirely.

```
procedure TForm2.TMSFNCWebSocketServer1Ping(Sender: TObject;
  AConnection: TTMSFNCWebSocketConnection; const aData: TArray<System.Byte>);
begin
  AConnection.SendSimpleFrame(focPing);
end;
```

The `OnPong` event is triggered whenever a pong message is received from a client.

2.1.6 Secure connection

Traffic over a websocket server can be protected via TLS. In order to enable this, set the `UseSSL` property to `True`.

The `CertificateFile`, `CertificateKeyFile` and `RootCertificateFile` properties can be used to define the path to the certificate file(s).

For secure connection OpenSSL libraries are required.

- `Source files`
- `Precompiled`

2.1.7 Properties

Property name	Description
Active: Boolean	Start or stop the server.
CertificateFile: string	Path to the certificate file. Used when `UseSSL` is set to `True`.
CertificateKeyFile: string	Path to the certificate key file. Used when `UseSSL` is set to `True`.
FrameSize: UIn64	The size of each frame to be sent. Used when `SplitMessage` is set to `True`.
Options: TTMSFNCWebsocketOptions	A set connection related options. Values are: - `twsoFrameByFrame`: Do not assemble the frames - `twsoSkipUpgradeCheck`: Do not check handshake Upgrade header - `twsoSkipVersionCheck`: Do not check handshake version - `twsoManualPong`: Do not send pong automatucally - `twsoManualClose`: Do not send close reply automatically
PathName: string	Name of the path. *For example*: ws://localhost:8888/**path**
Port: Integer	Port number. Default setting is `8888`.
RootCertificateFile: string	Path to the root certificate file. Used when `UseSSL` is set to `True`.
SplitMessage: Boolean	If set to `True`, messages are split into multiple frames based on `FrameSize`.
UseSSL: Boolean	If set to `True`, the server communication is protected via TLS.

2.1.8 Methods

Method name	Description
BroadcastMessage(AMessage: string)	Broadcasts a string message to all connected clients.
BroadcastData(AData: TBytes)	Broadcasts a binary message to all connected clients.
SendMessageTo(AMessage: string; ASelector: TTMSFNCWebSocketSendToCallback)	Sends a string message to connected clients. The `ASelector` callback determines to which clients the message should be sent.
SendDataTo(AData: TBytes; ASelector: TTMSFNCWebSocketSendToCallback)	Sends a binary message to connected clients. The `ASelector` callback determines to which clients the message should be sent.

2.1.9 Events

Event name	Description
OnAllow	Event triggered to allow or disallow a client connection.
OnBinaryDataReceived	Event triggered when a binary message arrives to the server.
OnClose	Event triggered when a close frame is sent by the client.
OnConnect	Event triggered when a clients *connects* to the server. This is not equal to when the client is connected (handshake successfully finished).
OnDisconnect	Event triggered when a client disconnects.
OnHandshakeResponseSent	Event triggered when the handshake response is sent to the client.
OnMessageReceived	Event triggered when a string message arrives to the server.
OnPing	Event triggered when a ping message arrives from a client.
OnPong	Event triggered when a pong message arrives from a client.

2.2 TTMSFNCWebSocketClient

The `TTMSFNCWebSocketClient` is a non-visual component enabling to perform websocket communication with a websocket server. It implements the RFC 6455 standard for websocket communication protocol.

> **Note**
>
> For WEB use the existing `TWebSocketClient` component.

2.2.1 Connect to a server

The basic steps to connect to a server are the following:

- Set the `HostName` property to the server host.
- Set the `Port` to the port of the websocket server.
- Set the `PathName`. **Optional**
  For example: in case of `ws://localhost:5050/socketpath` the `PathName` equals to `socketpath`.
- Set `Active` to `True` or call `Connect`.

2.2.2 Closing a connection

Calling `Disconnect` will send the closing frame to the server and wait for the reply. After receiving a closing frame in return, the socket client will close the underlying TCP connection.

Non-WEB only To abort/terminate the connection without sending the closing frame, pass a `False` boolean value as a parameter to the `Disconnect` call.

2.2.3 Send data/messages

Use `Send(AMessage: string)` or `Send(ABytes: TBytes)` to send a string or binary messages to the server.

```
procedure TForm1.messageBtnClick(Sender: TObject);
begin
  TMSFNCWebSocketClient1.Send('Hello server!');
end;
```

Send in multiple frames

By default all messages are sent in one single frame. If `SplitMessage` is set to `True`, the message will be split up into multiple frames based on size defined by `FrameSize`. Size of the frame includes frame headers too.

2.2.4 Receive data/messages

When a string message arrives from the server the `OnMessageReceived` event will be triggered. The `AMessage` parameter will contain the string message from the server.

When a binary message arrives from the server the `OnBinaryDataReceived` event will be triggered. The `AData` parameter will contain the binary message from the server.

2.2.5 Ping and pong

Whenever a ping message arrives from the server the `OnPing` event is triggered and an automatic pong message is sent back.

If the `Options` property contains the `twsoManualPong` value the poing message is **NOT** sent automatically. In this case the `OnPing` can be used to send manually or skip it entirely.

```
procedure TForm1.TMSFNCWebsocketClient1Ping(Sender: TObject;
  AConnection: TTMSFNCWebSocketConnection; const aData: TArray<System.Byte>);
begin
  AConnection.SendSimpleFrame(focPong);
end;
```

The `OnPong` event is triggered whenever a pong message is received from the server.

2.2.6 Secure connection

WEB clients

Connection thorugh browsers supported as:

- `http` -> `ws`, `wss`
- `https` -> `wss`

To force secure connection from an `http` source, set the `UseSSL` property to `True`.

Non-WEB clients

When connecting to a secure websocket server, the `UseSSL` property must be set to `True`.

For secure connection OpenSSL libraries are required.

- `Source files`
- `Precompiled`

2.2.7 Properties

Property name	Description
Active: Boolean	Set `True` to connect and `False` to disconnect.
AutoCheckInterval: Integer	Used when `AutoCheckMessages` is enabled.
AutoCheckMessages: Boolean	Check for incoming messages automatically. The interval of this action is defined by `AutoCheckInterval`.
CheckTimeout: Integer	Timeout for message check.
ConnectTimeout: Integer	Timeout for connection.
FrameSize: Uint64	The size of each frame to be sent. Used when `SplitMessage` is set to `True`.
HostName: string	Name of the host. *For example*: ws://**hostname**:port/path
Options: TTMSFNCWebsocketOptions	A set connection related options. Values are: - `twsoFrameByFrame` : Do not assemble the frames - `twsoSkipUpgradeCheck` : Do not check handshake Upgrade header - `twsoSkipVersionCheck` : Do not check handshake version - `twsoManualPong` : Do not send pong automatucally - `twsoManualClose` : Do not send close reply automatically
Origin: string	Optional header field for non-browser clients.
PathName: string	Name of the path. *For example*: ws://hostname:port/**path**
Port: Integer	Port number. *For example*: ws://hostname:**port**/path
SplitMessage: Boolean	If set to `True`, messages are split into multiple frames based on `FrameSize`.
UseSSL: Boolean	Set to `True` if the server is protected via TLS.

2.2.8 Methods

Method name	Description
CheckIncoming: TTMSFNCWebSocketIncomingResult	Manually trigger message reading.
Connect	Connect to a server and perform handshake
Disconnect(SendClose: Boolean = True)	Disconnect from the server. If `SendClose` is `False`, the client will not send a closing frame. If `True`, it will send a closing frame and wait for reply from the server.
Ping(AMessage: string)	Sends a ping frame to the server with an optional message.
Send(ABytes: TBytes)	Sends a string message to the server.
Send(AMessage: string)	Sends a binary message to the server

2.2.9 Events

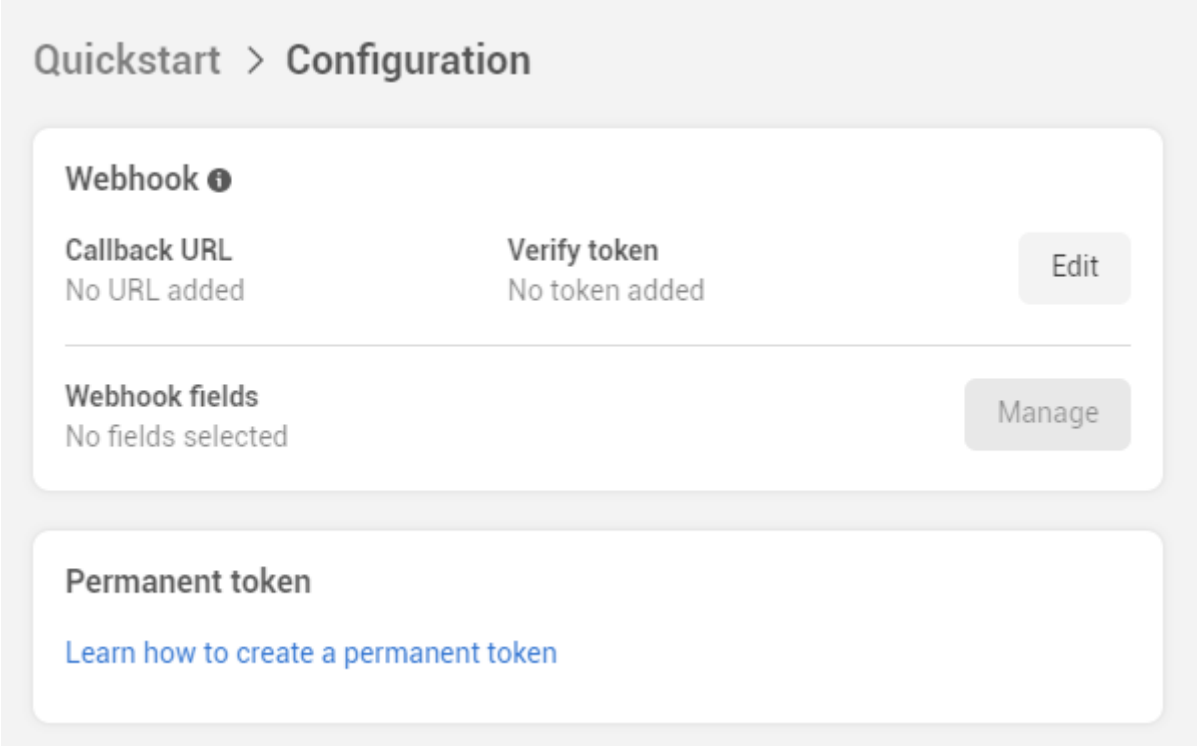
Event name	Description
OnBinaryDataReceived	Event triggered when binary data is received from the server.
OnConnect	Event triggered when connection to the server is successful.
OnDisconnect	Event triggered when client is disconnected from the server. Keep in mind this is event does not trigger if the connection is suddenly lost. To make sure you have connection to the server, use the `Ping` method to periodically check the connection.
OnHandshakeResponse	Event triggered when handshake is completed.
OnMessageReceived	Event triggered when string data is received from the server.
OnPing	Event triggered when a ping message is received from the server.
OnPong	Event triggered when a pong message is received from the server.
OnSendHandshake	Event triggered when handshake is sent.

2.3 TTMSFNCWhatsAppServer

The `TTMSFNCWhatsAppServer` is a non-visual component that can be used to host a server for the callback URL in the WhatsApp integration API. You can parse the incoming messages directly before the component automatically forwards them to the connected `TTMSFNCWhatsAppReceiver` clients. Under the hood it uses `TTMSFNCWebSocketServer`.

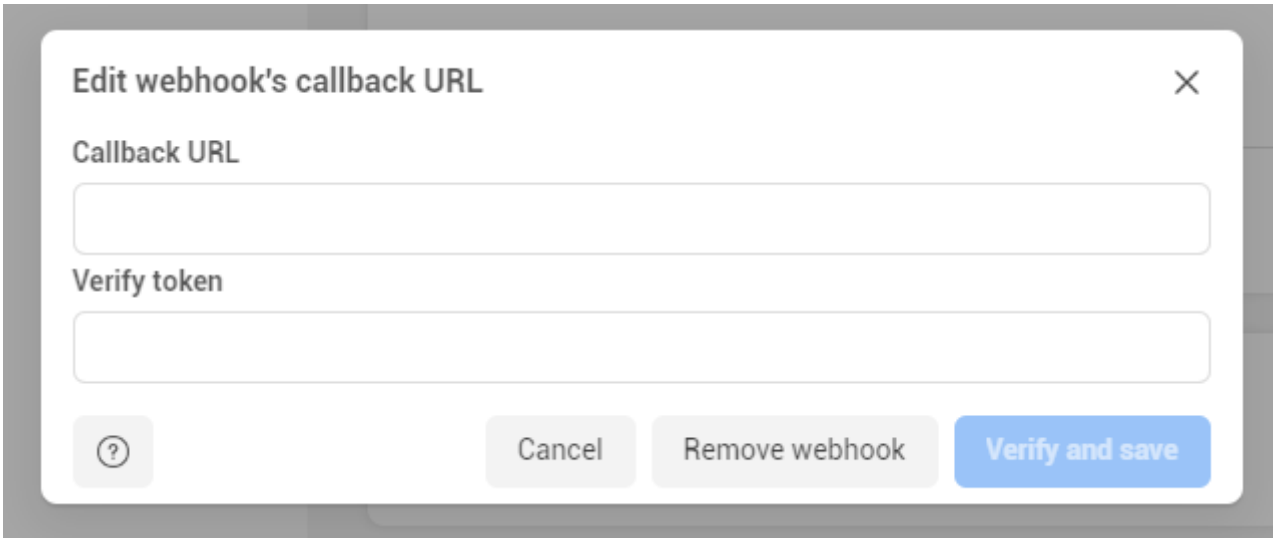
2.3.1 Setup

1. For initial setup of the account please follow the guide from `here`.
2. After setting up the application and adding the WhatsApp integration, you can set the callback URL. To do this, select `WhatsApp > Configuration` from the side menu in your application's dashboard.



3. Edit the Callback URL under Webhook. Set this value to a secure (HTTPS) URL, the same URL where your server is running. Fill in the Verify token during development with a string of your choice. The same verify token will need to be used in the `TTMSFNCWhatsAppServer`.

Later you can opt for a permanent token.



4. After the callback URL is successfully added, register for the fields of your choice. In the Webhook fields click on Manage and subscribe to `messages` to get notifications on incoming replies to the WhatsApp account.

You can subscribe to multiple fields, all of the messages will be received by the server. Click Done.

2.3.2 Forward messages to clients

All messages are forwarded to the connected `TTMSFNCWhatsAppReceiver` clients. However, before broadcasting the incoming messages, you can parse their value on server side and/or disable the broadcasting by implementing the `OnRawMessage` event.

```
procedure TForm1.TMSFNCWhatsAppServer1RawMessage(Sender: TObject;
  AMessage: string; var ABroadcast: Boolean);
begin
  //AMessage is a JSON string. Parse here if necessary.
  //ABroadcast := False will cause the message not to be sent to the connected clients
end;
```

2.3.3 Secure connection

This server must always run from a secure context `per WhatsApp requirements`. Because of this `UseSSL` is set to `True` and cannot be modified. See the `Secure connection` part of `TTMSFNCWebSocketServer`.

2.3.4 Properties

Property name	Description
Active: Boolean	Start or stop the server.
CertificateFile: string	Path to the certificate file.
CertificateKeyFile: string	Path to the certificate key file.
PathName: string	Name of the path. *For example*: https://localhost:8888/**path**
Port: Integer	Port number.
RootCertificateFile: string	Path to the root certificate file.
VerifyToken: string	It should be the same value as used during the `setup` for the Verify token.

2.3.5 Methods

Method name	Description
BroadcastMessage(AMessage: string)	Broadcasts a string message to all connected clients.
BroadcastData(AData: TBytes)	Broadcasts a binary message to all connected clients.
SendMessageTo(AMessage: string; ASelector: TTMSFNCWebSocketSendToCallback)	Sends a string message to connected clients. The `ASelector` callback determines to which clients the message should be sent.
SendDataTo(AData: TBytes; ASelector: TTMSFNCWebSocketSendToCallback)	Sends a binary message to connected clients. The `ASelector` callback determines to which clients the message should be sent.

2.3.6 Events

Event name	Description
OnAllow	Event triggered when a new client tries to connect.
OnBinaryDataReceived	Event triggered when binary message arrives from the connected clients.
OnClose	Event triggered when a close frame is sent by the client.
OnCommandGet	Event triggered when an HTTP request arrives.
OnConnect	Event triggered when a clients connects.
OnDisconnect	Event triggered when a client disconnects.
OnGetSSLPassword	Event triggered when the pass phrase is needed that was mentioned while creating the SLL certificate.
OnMessageReceived	Event triggered when a string message arrives from the connected clients.
OnPing	Event triggered when a ping message arrives from a client.
OnPong	Event triggered when a pong message arrives from a client.
OnRawMessage	Event triggered when a message arrives from WhatsApp. The `AMessage` string parameter is in JSON format.

2.4 TTMSFNCWhatsAppReceiver

The `TTMSFNCWhatsAppReceiver` is a non-visual component to receive messages sent to a WhatsApp account. For this component to receive WhatsApp messages it must connect to a server hosted by `TTMSFNCWhatsAppServer`, running on a secure context. It uses `TTMSFNCWebSocketClient` internally.

2.4.1 Receive messages

Whenever a message arrives from `TTMSFNCWhatsAppServer` the `OnBinaryDataReceived` / `OnMessageReceived` event is triggered first, depending on the message format. After that in case of a string message it is further parsed internally into a record that contains all the WhatsApp message related information. Any incoming messages that don't follow the `messages` field format will be discarded.

If a valid formatted message arrives the `OnWhatsAppMessageReceived` event will be triggered. The `AMessage` parameter contains the message details, such as sender name, phone number, message type, and so on...

2.4.2 Secure connection

The `TTMSFNCWhatsAppServer` is always required to run in a secure context. This means the `UseSSL` is set to `True` and cannot be modified. See the `Secure connection` part of `TTMSFNCWebSocketClient`.

2.4.3 Properties

Property name	Description
Active: Boolean	Set `True` to connect and `False` to disconnect.
HostName: string	Name of the host.
PathName: string	Name of the path. *For example*: https://hostname:port/**path**
Port: Integer	Port number.

2.4.4 Methods

Method name	Description
Connect	Connect to a server and perform handshake.
Disconnect(SendClose: Boolean = True)	Disconnect from the server. - Non-WEB The `SendClose` parameter is for non-WEB platforms only. If `SendClose` is `False`, the client will not send a closing frame. If `True`, it will send a closing frame and wait for reply from the server.
Send(ABytes: TBytes)	Send a string message to the server.
Send(AMessage: string)	Send a binary message to the server.
Ping(AMessage: string) Non-WEB only	Sends a ping frame to the server with an optional message.

2.4.5 Events

Event name	Description
OnBinaryDataReceived	Event triggered when binary data is received from the server.
OnConnect	Event triggered when connection to the server is successful.
OnDisconnect	Event triggered when client is disconnected from the server. - WEB Triggered automatically whenever the connection is lost. - Non-WEB Does not trigger if the connection is suddenly lost. To make sure you have connection to the server, use the `Ping` method to periodically check the connection.
OnMessageReceived	Event triggered when a string message arrives from the server. This includes both forwarded messages from the callback URL and the `BroadcastMessage` / `SendMessageTo` calls by the server.
OnPing Non-WEB only	Event triggered when a ping message is received from the server.
OnPong Non-WEB only	Event triggered when a pong message is received from the server.
OnWhatsAppMessageReceived	Event triggered when a valid `message` field formatted message arrives from the server.