SGCWEBSOCKETS · TECHNICAL DOCUMENT

3Commas API

3Commas trading-bot WebSocket and REST client for Delphi — DMC streams plus account, deal and bot endpoints.

eSeGeCe

WWW.ESEGECE.COM

Overview

The websocket feed provides real-time market data updates for Trades and Deals

At a glance

COMPONENT CLASS

`TsgcWSAPI_ThreeCommas`

STANDARDS / SPEC

3Commas API documentation

TRANSPORTS

TCP, TLS

PLATFORMS

Windows, macOS, Linux, iOS, Android

FRAMEWORKS

VCL, FireMonkey, Lazarus / FPC, .NET

EDITION

Standard / Professional / Enterprise

Features

- Native Delphi implementation with full ANSI/Unicode support.

Technical specification

Standards & specs	3Commas API documentation
Component class	`TsgcWSAPI_ThreeCommas` (unit `sgcWebSocket_API_ThreeCommas`)
Frameworks	VCL, FireMonkey, Lazarus / FPC, .NET
Platforms	Windows, macOS, Linux, iOS, Android

Main properties

The principal published / public properties used to configure and drive the component. Consult the online help for the full list.

`Client`	Published or public property used to configure or query the component.
`OnConnect`	Published or public property used to configure or query the component.
`OnThreeCommasConnect`	Published or public property used to configure or query the component.
`OnThreeCommasConfirmSubscription`	Published or public property used to configure or query the component.
`OnThreeCommasRejectSubscription`	Published or public property used to configure or query the component.
`OnThreeCommasMessage`	Published or public property used to configure or query the component.
`OnThreeCommasPing`	Published or public property used to configure or query the component.
`OnThreeCommasHTTPException`	Published or public property used to configure or query the component.
`OnDisconnect`	Published or public property used to configure or query the component.

`ThreeCommas`	Published or public property used to configure or query the component.

Main methods

The principal public methods exposed by the component.

`SubscribeSmartTrades()`	Public procedure exposed by the component.
`SubscribeDeals()`	Public procedure exposed by the component.

Quick Start

Drop the component on a form, configure the properties below and activate it. The snippet that follows shows the typical **Send Commands** configuration sourced from the online help.

> **About this scenario.** Use the method ExecCommand to send commands to the server. The responses will be available OnResponse Event.

Delphi (VCL / FireMonkey)

```
oRCON := TsgcLib_RCON.Create(nil);
oRCON.RCON_Options.Host := '127.0.0.1';
oRCON.RCON_Options.Port := 25575;
oRCON.RCON_Options.Password := 'test';
oRCON.Active := True;
<br/>

procedure OnAuthenticate(Sender: TObject; Authenticated: Boolean; const aPacket: TsgcRCON_Packet
begin
if Authenticated then
DoLog('#authenticated')
else
DoLog('#not authenticated');
end;
<br/>

procedure OnResponse(Sender: TObject; const aResponse: string; const aPacket: TsgcRCON_Packet);
begin
DoLog(aResponse);
end;
```

C++ Builder

```
TsgcLib_RCON oRCON = new TsgcLib_RCON();
oRCON->RCON_Options->Host = "127.0.0.1";
oRCON->RCON_Options->Port = 25575;
oRCON->RCON_Options->Password = "test";
oRCON->Active = true;
<br/>

void OnAuthenticate(TObject *Sender, bool Authenticated, const TsgcRCON_Packet *aPacket)
{
if (Authenticated == true)
{
DoLog("#authenticated");
}
else
{
DoLog("#not authenticated");
}
}
<br/>

void OnResponse(Object *Sender, const string aResponse, const TsgcRCON_Packet *aPacket)
{
DoLog(aResponse);
}
```

.NET (C#)

```
TsgcLib_RCON oRCON = new TsgcLib_RCON();
oRCON.RCON_Options.Host = "127.0.0.1";
oRCON.RCON_Options.Port = 25575;
oRCON.RCON_Options.Password = "test";
oRCON.Active = true;
<br/>

void OnAuthenticate(TObject Sender, bool Authenticated, const TsgcRCON_Packet aPacket)
{
if (Authenticated == true)
{
DoLog("#authenticated");
}
else
{
DoLog("#not authenticated");
}
}
<br/>

void OnResponse(Object Sender, const string aResponse, const TsgcRCON_Packet aPacket)
{
DoLog(aResponse);
}
```

Common scenarios

The following scenarios are lifted verbatim from the online help. Each shows the configuration and method calls needed to drive the component through a specific real-world flow.

1 · Telegram | Send Bot Message With Buttons

Telegram API allows you to send messages with buttons to request data from the user (this option is only available for bots).

Delphi (VCL / FireMonkey)

```
oReplyMarkup := TsgcTelegramReplyMarkupShowKeyboard.Create;
Try
oReplyMarkup.AddButtonTypeRequestPhoneNumber('Give me your phone');
sgcTelegram.SendTextMessage('123456', 'Please provide the information below', nil, oReplyMarkup)
Finally
oReplyMarkup.Free;
End;
```

C++ Builder

```
oReplyMarkup = new TsgcTelegramReplyMarkupShowKeyboard();
oReplyMarkup->AddButtonTypeRequestPhoneNumber("Give me your phone");
sgcTelegram->SendTextMessage("123456", "Please provide the information below", null, oReplyMarku
oReplyMarkup->Free();
```

.NET (C#)

```
oReplyMarkup = new TsgcTelegramReplyMarkupShowKeyboard();
oReplyMarkup.AddButtonTypeRequestPhoneNumber("Give me your phone");
sgcTelegram.SendTextMessage("123456", "Please provide the information below", null, oReplyMarkup
```

2 · Telegram | Send Telegram Invoice Message

If your bot supports inline mode, users can also send invoices to other chats via the bot, including to one-on-one chats with other users.

Delphi (VCL / FireMonkey)

```
procedure SendInvoice;
var
  oInvoice: TsgcTelegramSendInvoice;
begin
  oInvoice := TsgcTelegramSendInvoice.Create;
  Try
    oInvoice.Title := 'Invoice Title Test';
    oInvoice.Description := 'Description Invoice Test';
    oInvoice.Invoice.Currency := 'EUR';
    oInvoice.Invoice.Total := 800;
    oInvoice.Invoice.IsTest := True;
        oInvoice.Invoice.Payload := 'payload';
        oInvoice.Invoice.ProviderToken := 'provider_token';
        oInvoice.Invoice.ProviderData := 'provider_data';

    sgcTelegram.SendInvoiceMessage('3284239872', oInvoice);
  Finally
    oInvoice.Free;
  End;
end;
```

C++ Builder

```
private void SendInvoice()
{
  TsgcTelegramSendInvoice *oInvoice = new TsgcTelegramSendInvoice();
  Try
  {
    oInvoice->Title = "Invoice Title Test";
    oInvoice->Description = "Description Invoice Test";
    oInvoice->Invoice->Currency = 'EUR';
    oInvoice->Invoice->Total = 800;
    oInvoice->Invoice->IsTest = True;
        oInvoice->Invoice->Payload := "payload";
        oInvoice->Invoice->ProviderToken := "provider_token";
        oInvoice->Invoice->ProviderData := "provider_data";

    sgcTelegram->SendInvoiceMessage("3284239872", oInvoice);
  __finally
  {
    oInvoice->Free();
  }
}
```

.NET (C#)

```
private void SendInvoice()
{
  TsgcTelegramSendInvoice oInvoice = new TsgcTelegramSendInvoice();
  oInvoice.Title = 'Invoice Title Test';
  oInvoice.Description = 'Description Invoice Test';
  oInvoice.Invoice.Currency = 'EUR';
  oInvoice.Invoice.Total = 800;
  oInvoice.Invoice.IsTest = True;
  oInvoice.Invoice.Payload := "payload";
  oInvoice.Invoice.ProviderToken := "provider_token";
  oInvoice.Invoice.ProviderData := "provider_data";

  sgcTelegram.SendInvoiceMessage("3284239872", oInvoice);
}
```

3 · Telegram | Send Telegram Message With Inline Buttons

Telegram API allows you to send messages with inline buttons to select options as an answer (this option is only available for bots).

Delphi (VCL / FireMonkey)

```
oReplyMarkup := TsgcTelegramReplyMarkupInlineKeyboard.Create;
Try
  oReplyMarkup.AddButtonTypeCallback('Yes', 'I like it');
  oReplyMarkup.AddButtonTypeCallback('No', 'I hate it');
  oReplyMarkup.AddButtonTypeUrl('Poll', 'https://www.yoursite.com/telegram/poll');
  sgcTelegram.SendTextMessage('123456', 'Do you like the message?', oReplyMarkup);
Finally
  oReplyMarkup.Free;
End;

      procedure OnNewCallbackQuery(Sender: TObject; CallbackQuery: TsgcTelegramCallbackQuery);
begin
  if CallbackQuery.PayloadData.Data = 'I like it' then
    ShowMessage('yes')
  else
    ShowMessage('no');
end;
```

C++ Builder

```
TsgcTelegramReplyMarkupInlineKeyboard *oReplyMarkup = new TsgcTelegramReplyMarkupInlineKeyboard(
try
{
  oReplyMarkup->AddButtonTypeCallback("Yes", "I like it");
  oReplyMarkup->AddButtonTypeCallback("No", "I hate it");
  oReplyMarkup->AddButtonTypeUrl("Poll", "https://www.yoursite.com/telegram/poll");
  sgcTelegram->SendTextMessage("123456", "Do you like the message?", oReplyMarkup);
}
__finally
{
  oReplyMarkup->Free();
}

void OnNewCallbackQuery(TObject *Sender, TsgcTelegramCallbackQuery *CallbackQuery)
{
  if (CallbackQuery->PayloadData->Data == "I like it") then
  {
    ShowMessage("yes")
  }
  else
  {
    ShowMessage("no");
  }
}
```

.NET (C#)

```
TsgcTelegramReplyMarkupInlineKeyboard oReplyMarkup = new TsgcTelegramReplyMarkupInlineKeyboard()
oReplyMarkup.AddButtonTypeCallback("Yes", "I like it");
oReplyMarkup.AddButtonTypeCallback("No", "I hate it");
oReplyMarkup.AddButtonTypeUrl("Poll", "https://www.yoursite.com/telegram/poll");
sgcTelegram.SendTextMessage("123456", "Do you like the message?", oReplyMarkup);

void OnNewCallbackQuery(TObject Sender, TsgcTelegramCallbackQuery CallbackQuery)
{
  if (CallbackQuery.PayloadData.Data == "I like it") then
  {
    MessageBox.Show("yes")
  }
  else
  {
    MessageBox.Show("no");
  }
}
```

4 · Get Started

To send and receive a first message using a test number, complete the following steps:

Delphi (VCL / FireMonkey)

```
oClient := TsgcWhatsapp_Client.Create(nil);
oClient.WhatsappOptions.PhoneNumberId := '107809351952205';
oClient.WhatsappOptions.Token := 'EAAO4OpgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2';
oClient.SendTest('34605889421');
```

C++ Builder

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAAO4OpgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->SendTest("34605889421");
```

.NET (C#)

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient.WhatsappOptions.PhoneNumberId = "107809351952205";
oClient.WhatsappOptions.Token = "EAAO4OpgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient.SendTest("34605889421");
```

5 · Received Messages

Every time a new message is received the event OnMessageReceived is called, where you can access to the content of the Message and mark the message as read.

Delphi (VCL / FireMonkey)

```
procedure OnWhatsAppMessageReceived(Sender: TObject; const aMessage: TsgcWhatsApp_Receive_Messag
var
  vText: string;
  vTo: string;
begin
  if aMessage.Contacts.Count > 0 then
  begin
    vTo := aMessage.Contacts.Contact[0].WaID;
    if aMessage.Messages.Count > 0 then
    begin
      if aMessage.Messages._Message[0]._Type = wapmrtText then
      begin
        vText := 'ECHO ⟹ ' + aMessage.Messages._Message[0].Text.Body;
        WhatsApp.SendMessageText(vTo, vText);
        aMarkAsRead := True;
      end;
    end;
  end;
end;
```

C++ Builder

```
void OnWhatsAppMessageReceived(TObject *Sender, const TsgcWhatsApp_Receive_Message *aMessage, re
{
  if (aMessage->Contacts->Count > 0)
  {
    string vTo = aMessage->Contacts->Contact[0]->WaID;
    if (aMessage->Messages->Count > 0)
    {
      if (aMessage->Messages->_Message[0]->_Type = wapmrtText)
      {
        vText = "ECHO ⟹ " + aMessage->Messages->_Message[0]->Text->Body;
        WhatsApp->SendMessageText(vTo, vText);
        aMarkAsRead = true;
      }
    }
  }
}
```

.NET (C#)

```
void OnWhatsAppMessageReceived(TsgcWhatsApp_Client Sender, TsgcWhatsApp_Receive_Message Message,
{
  DoLog("Message Received: [" + Message.From + "] " + Message.Text);
 MarkAsRead = true;
}
```

6 · Add Proxy

In order to configure a HTTP Proxy, first you must add the proxy to telegram configuration, to do this, just call AddProxyHTTP and if successful, a message will be returned with the new proxy added. Once the proxy has been added to the list, just call EnableProxy and pass the ID of the proxy received on the confirmation message.

Delphi (VCL / FireMonkey)

```
Telegram.AddProxyHTTP('8.8.8.8', 8080, '', '', True);
// ... read the confirmation message and save the ID of the proxy.
Telegram.EnableProxy(2);
```

C++ Builder

```
Telegram->AddProxyHTTP("8.8.8.8", 8080, "", "", true);
// ... read the confirmation message and save the ID of the proxy.
Telegram->EnableProxy(2);
```

.NET (C#)

```
Telegram.AddProxyHTTP("8.8.8.8", 8080, "", "", true);
// ... read the confirmation message and save the ID of the proxy.
Telegram.EnableProxy(2);
```

Sources used to build this document

Every external claim links back to a primary source. The online-help references decode the canonical deep-link the company maintains for this component.

Primary standard / spec — 3Commas API documentation	developers.3commas.io/
Online help — component page	www.esegece.com/help/sgcWebSockets/Components/APIs/API/API_3Commas.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\05.Crypto\10.ThreeCommas`
.NET demo project (in the sgcWebSockets package)	`.net\demos\05.Crypto\10.ThreeCommas`
Component page	www.esegece.com/products/websockets/apis/3commas/
Product page	www.esegece.com/products/websockets/

> **Document scope.** This document covers the publicly-documented surface of the 3Commas API component shipped with sgcWebSockets. For full property, method and event reference consult the online help linked above.