

AI ChatBot

High-level ChatBot component — wraps OpenAI, Anthropic, Gemini or local LLMs in a one-line conversational front-end.

Overview

The microphone audio must be captured, so a speech-to-text system is needed to get the text that will be sent to OpenAI.

At a glance

COMPONENT CLASS

TsgcAI0penAIChatBot

STANDARDS / SPEC

OpenAI Chat Completions

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	OpenAI Chat Completions · Anthropic Messages
Component class	<code>TsgcAIOpenAIChatBot</code> (unit <code>sgcAI_Chat</code>)
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.

<code>OpenAIOptions</code>	OpenAI API credentials and logging (ApiKey, Organization, LogOptions) shared by Whisper, ChatCompletion and Embeddings calls.
<code>ChatBotOptions</code>	ChatBot behaviour: Transcription (Whisper model/language) and ChatCompletion (model, Enabled flag).
<code>TextToSpeech</code>	Component used to speak the assistant reply (System, Google or Amazon TTS).
<code>AudioRecorder</code>	Component used to capture microphone audio that is sent to Whisper for transcription.
<code>Embeddings</code>	Optional embeddings store used to retrieve context (RAG) before a chat completion.
<code>Version</code>	Read-only sgcWebSockets library version string.

Main methods

The principal public methods exposed by the component.

<code>Start()</code>	Starts microphone recording through the attached AudioRecorder.
<code>Stop()</code>	Stops recording; the captured audio is then sent to Whisper for transcription.

ChatAsUser()

Sends a text prompt as the user role and triggers a ChatCompletion reply (spoken via TextToSpeech).

ChatAsSystem()

Sends a system-role instruction to steer the assistant; speech output is off by default.

Public events

The component exposes the following published events; consult the online help for full event-handler signatures.

OnAudioStart

Fires when the AudioRecorder begins capturing microphone audio.

OnAudioStop

Fires after microphone capture stops and before transcription is sent to Whisper.

OnChatCompletion

Fires when the OpenAI ChatCompletion API returns an assistant reply; exposes Role and Content.

OnTranscription

Fires when Whisper returns the speech-to-text result; lets you edit the text or reject it.

Quick Start

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

About this scenario. Once we've converted all our data to vectors, we can start to build our own model. The idea behind it is very simple: every time we ask the bot, first we convert the question to a vector, then we search our database for which vector is most similar to the question, and finally we use the most similar data and add it as context.

Delphi (VCL / FireMonkey)

```
procedure AskToChatGPT(const aQuestion: string);
var
  oChatBot: TsgcAIOpenAIChatBot;
  oEmbeddings: TsgcAIOpenAIEmbeddings;
  oFile: TsgcAIDatabaseVectorFile;
  vContext: string;
begin
  oChatBot := TsgcAIOpenAIChatBot.Create(nil);
  Try
    oChatBot.OpenAIOptions.ApiKey := '<your api key>';
    oEmbeddings := TsgcAIOpenAIEmbeddings.Create(nil);
  Try
    oChatBot.Embeddings := oEmbeddings;
    oFile := TsgcAIDatabaseVectorFile.Create(nil);
  Try
    oEmbeddings.Database := oFile;
    vContext := oChatBot.GetEmbedding(aQuestion);
    oChatBot.ChatAsUser('Answer the question based on the context below.\n\nContext:\n' +
      vContext + '\nQuestion:' + aQuestion + '\nAnswer:');
  Finally
    oFile.Free;
  End;
  Finally
    oEmbeddings.Free;
  End;
  Finally
    FreeAndNil(oDialog);
  End;
end; </code>
<code class="delphi">
```

C++ Builder

```
void AskToChatGPT(const std::string& aQuestion)
{
    TsgcAIOpenAIChatBot* oChatBot = new TsgcAIOpenAIChatBot(NULL);
    try
    {
        oChatBot->OpenAIOptions->ApiKey = "<your api key>";
        TsgcAIOpenAIEmbeddings* oEmbeddings = new TsgcAIOpenAIEmbeddings(NULL);
        try
        {
            oChatBot->Embeddings = oEmbeddings;
            TsgcAIDatabaseVectorFile* oFile = new TsgcAIDatabaseVectorFile(NULL);
            try
            {
                oEmbeddings->Database = oFile;
                std::string vContext = oChatBot->GetEmbedding(aQuestion);
                std::string message = "Answer the question based on the context below.\n\nContext:\n" +
                    vContext + "\nQuestion:" + aQuestion + "\nAnswer:";
                oChatBot->ChatAsUser(message.c_str());
            }
            __finally
            {
                delete oFile;
            }
        }
        __finally
        {
            delete oEmbeddings;
        }
    }
    __finally
    {
        delete oChatBot;
    }
}
```

.NET (C#)

```
public void AskToChatGPT(string aQuestion)
{
    using (TsgcAIOpenAIChatBot oChatBot = new TsgcAIOpenAIChatBot())
    {
        oChatBot.OpenAIOptions.ApiKey = "<your api key>";
        using (TsgcAIOpenAIEmbeddings oEmbeddings = new TsgcAIOpenAIEmbeddings())
        {
            oChatBot.Embeddings = oEmbeddings;
            using (TsgcAIDatabaseVectorFile oFile = new TsgcAIDatabaseVectorFile())
            {
                oEmbeddings.Database = oFile;
                string vContext = oChatBot.GetEmbedding(aQuestion);
                string message = "Answer the question based on the context below.\n\nContext:\n" +
                    vContext + "\nQuestion:" + aQuestion + "\nAnswer:";
                oChatBot.ChatAsUser(message);
            }
        }
    }
}
```

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 · Embeddings | Create Vectors

To use the embeddings, first we must convert our data to vectors.

Delphi (VCL / FireMonkey)

```
procedure ConvertFileToVector;
var
  oDialog: TOpenDialog;
  oEmbeddings: TsgcAIOpenAIEmbeddings;
  oFile: TsgcAIDatabaseVectorFile;
begin
  oDialog := TOpenDialog.Create(nil);
  Try
    oDialog.Filter := 'TXT Files|*.txt';
    if oDialog.Execute then
      begin
        oEmbeddings := TsgcAIOpenAIEmbeddings.Create(nil);
        Try
          oFile := TsgcAIDatabaseVectorFile.Create(nil);
          Try
            oEmbeddings.Database := oFile;
            oEmbeddings.OpenAIOptions.ApiKey := '<your api key>';
            oEmbeddings.CreateEmbeddingsFromFile(oDialog.FileName);
          Finally
            oFile.Free;
          End;
        Finally
          oEmbeddings.Free;
        End;
      end;
  Finally
    FreeAndNil(oDialog);
  End;
end;
```

C++ Builder

```

void ConvertFileToVector()
{
    TOpenDialog* oDialog = new TOpenDialog(NULL);
    try
    {
        oDialog->Filter = "TXT Files|*.txt";
        if (oDialog->Execute())
        {
            TsgcAIOpenAIEmbeddings* oEmbeddings = new TsgcAIOpenAIEmbeddings(NULL);
            try
            {
                TsgcAIDatabaseVectorFile* oFile = new TsgcAIDatabaseVectorFile(NULL);
                try
                {
                    oEmbeddings->Database = oFile;
                    oEmbeddings->OpenAIOptions->ApiKey = "<your api key>";
                    oEmbeddings->CreateEmbeddingsFromFile(oDialog->FileName);
                }
                __finally
                {
                    delete oFile;
                }
            }
            __finally
            {
                delete oEmbeddings;
            }
        }
    }
    __finally
    {
        delete oDialog;
    }
}

```

.NET (C#)

```

public void ConvertFileToVector()
{
    using (OpenFileDialog oDialog = new OpenFileDialog())
    {
        oDialog.Filter = "TXT Files|*.txt";
        if (oDialog.ShowDialog() == DialogResult.OK)
        {
            using (TsgcAIOpenAIEmbeddings oEmbeddings = new TsgcAIOpenAIEmbeddings())
            {
                using (TsgcAIDatabaseVectorFile oFile = new TsgcAIDatabaseVectorFile())
                {
                    oEmbeddings.Database = oFile;
                    oEmbeddings.OpenAIOptions.ApiKey = "<your api key>";
                    oEmbeddings.CreateEmbeddingsFromFile(oDialog.FileName);
                }
            }
        }
    }
}

```

2 · Step 1 Define Functions

When creating your assistant, you will first define the functions under the tools param of the assistant.

Delphi (VCL / FireMonkey)

```

Assistant := TsgcAIOpenAIAssistant.Create(nil);
Assistant.OpenAIOptions.ApiKey := 'sk-askdjfalaskdjfl23kjkjasdefasdfj';
Assistant.AssistantOptions.Name := 'DeLphi Weather Bot';
Assistant.AssistantOptions.Instructions.Text := 'You are a weather bot. Use the provided functio
Assistant.AssistantOptions.Model := 'gpt-4o';
Assistant.AssistantOptions.Tools.Functions.Enabled := False;
Assistant.AssistantOptions.Tools.Functions.Functions.Text := '[{"type":"function","function":{"n
Assistant.AssistantOptions.Tools.FileSearch.Enabled := False;
Assistant.AssistantOptions.Tools.CodeInterpreter.Enabled := False;

```

C++ Builder

```
TsgcAIOpenAIAssistant *Assistant = new TsgcAIOpenAIAssistant(nullptr);
Assistant->OpenAIOptions->ApiKey = "sk-askdjfalaskdjfl23kjkjasdefasdfj";
Assistant->AssistantOptions->Name = "Delphi Weather Bot";
Assistant->AssistantOptions->Instructions->Text = "You are a weather bot. Use the provided funct
Assistant->AssistantOptions->Model = "gpt-4o";
Assistant->AssistantOptions->Tools->Functions->Enabled = false;
Assistant->AssistantOptions->Tools->Functions->Functions->Text =
    "[{\"type\":\"function\",\"function\":{\"name\":\"get_current_temperature\",\"description\":
Assistant->AssistantOptions->Tools->FileSearch->Enabled = false;
Assistant->AssistantOptions->Tools->CodeInterpreter->Enabled = false;
```

.NET (C#)

```
var assistant = new TsgcAIOpenAIAssistant(null);
assistant.OpenAIOptions.ApiKey = "sk-askdjfalaskdjfl23kjkjasdefasdfj";
assistant.AssistantOptions.Name = "Delphi Weather Bot";
assistant.AssistantOptions.Instructions.Text = "You are a weather bot. Use the provided function
assistant.AssistantOptions.Model = "gpt-4o";
assistant.AssistantOptions.Tools.Functions.Enabled = false;
assistant.AssistantOptions.Tools.Functions.Functions.Text = "[{\"type\":\"function\",\"function\"
assistant.AssistantOptions.Tools.FileSearch.Enabled = false;
assistant.AssistantOptions.Tools.CodeInterpreter.Enabled = false;
```

3 · Step 1: Create a new Assistant with File Search Enabled

Create a new assistant with `file_search` enabled in the `tools` parameter of the Assistant. Once the `file_search` tool is enabled, the model decides when to retrieve content based on user messages.

Delphi (VCL / FireMonkey)

```
Assistant := TsgcAIOpenAIAssistant.Create(nil);
Assistant.OpenAIOptions.ApiKey := 'sk-askdjfalaskdjfl23kjkjasdefasdfj';
Assistant.AssistantOptions.Name := 'sgcWebSockets HelpDesk';
Assistant.AssistantOptions.Instructions.Text := 'You are a sgcWebSockets HelpDesk Agent. ' +
'Answer questions briefly, in a sentence or less. When asked a question,use the manual to answer
Assistant.AssistantOptions.Model := 'gpt-4o-mini';
Assistant.AssistantOptions.Tools.FileSearch.Enabled := True;
Assistant.AssistantOptions.Tools.CodeInterpreter.Enabled := False;
```

C++ Builder

```
TsgcAIOpenAIAssistant* Assistant = new TsgcAIOpenAIAssistant(nullptr);
Assistant->OpenAIOptions->ApiKey = "sk-askdjfalaskdjfl23kjkjasdefasdfj";
Assistant->AssistantOptions->Name = "sgcWebSockets HelpDesk";
Assistant->AssistantOptions->Instructions->Text =
    "You are a sgcWebSockets HelpDesk Agent. "
    "Answer questions briefly, in a sentence or less. When asked a question, use the manual to a
Assistant->AssistantOptions->Model = "gpt-4o-mini";
Assistant->AssistantOptions->Tools->FileSearch->Enabled = true;
Assistant->AssistantOptions->Tools->CodeInterpreter->Enabled = false;
```

.NET (C#)

```
var Assistant = new TsgcAIOpenAIAssistant(null);
Assistant.OpenAIOptions.ApiKey = "sk-askdjfalaskdjfl23kjkjasdefasdfj";
Assistant.AssistantOptions.Name = "sgcWebSockets HelpDesk";
Assistant.AssistantOptions.Instructions.Text =
    "You are a sgcWebSockets HelpDesk Agent. " +
    "Answer questions briefly, in a sentence or less. When asked a question, use the manual to a
Assistant.AssistantOptions.Model = "gpt-4o-mini";
Assistant.AssistantOptions.Tools.FileSearch.Enabled = true;
Assistant.AssistantOptions.Tools.CodeInterpreter.Enabled = false;
```

4 · Step 4: Handle the Response

Use the event `OnStreamMessageDelta` to read the server-sent stream message.

Delphi (VCL / FireMonkey)

```
procedure OnStreamMessageDelta(Sender: TObject; const aMessageDelta: TsgcOpenAIClass_MessageDelta;
var
    i: Integer;
    vResponse: string;
    vType: string;
begin
    for i := Low(aMessageDelta.Content) to High(aMessageDelta.Content) do
        begin
            vType := aMessageDelta.Content[i]._Type;
            if vType = 'text' then
                begin
                    vResponse := TsgcOpenAIClass_MessageDeltaContent_Text
                        (aMessageDelta.Content[i]).Value;
                end;
        end;
end;
```

C++ Builder

```
void __fastcall OnStreamMessageDelta(TObject *Sender, const TsgcOpenAIClass_MessageDelta &aMessageDelta)
{
    for (int i = aMessageDelta.Content.Low(); i ≤ aMessageDelta.Content.High(); i++)
    {
        String vType = aMessageDelta.Content[i]→_Type;
        if (vType = "text")
        {
            String vResponse = static_cast<TsgcOpenAIClass_MessageDeltaContent_Text*>(aMessageDelta.Co
        }
    }
}
```

.NET (C#)

```
void OnStreamMessageDelta(object sender, TsgcOpenAIClass_MessageDelta aMessageDelta, string aRaw)
{
    foreach (var content in aMessageDelta.Content)
    {
        string vType = content._Type;
        if (vType = "text")
        {
            string vResponse = ((TsgcOpenAIClass_MessageDeltaContent_Text)content).Value;
        }
    }
}
```

5 · Step 2: Create a Thread and add Messages

Create a Thread when a user starts a conversation and add Messages to the Thread as the user asks questions.

Delphi (VCL / FireMonkey)

```

procedure SendMessage()
var
  i: Integer;
  oMessage: TsgcOpenAIClass_Message;
  oMessages: TsgcOpenAIClass_Response_List_Messages;
  oRun: TsgcOpenAIClass_Run;
begin
  DoLog('[user]: ' + memoMessage.Lines.Text);
  Screen.Cursor := crHourGlass;
  Try
    oMessage := Assistant.CreateMessageText('thread_id', 'What is the weather in San Francisco t
    if Assigned(oMessage) then
      begin
        oRun := Assistant.CreateRunAndWait('thread_id');
        if Assigned(oRun) then
          begin
            oMessages := Assistant.GetMessages('thread_id', oRun.Id);
            if Assigned(oMessages) and (Length(oMessages.Messages) > 0) then
              begin
                memoMessage.Lines.Text := '';
                for i := 0 to Length(oMessages.Messages) - 1 do
                  DoLog('[assistant]: ' + DoFormatResponse(oMessages.Messages[i]
                    .ContentText + #13#10));
                end;
              end;
            end;
          end;
        Finally
          Screen.Cursor := crDefault;
        End;
      end;
    end;
  end;
end;

```

C++ Builder

```

void SendMessage()
{
    int i;
    TsgcOpenAIClass_Message* oMessage = nullptr;
    TsgcOpenAIClass_Response_List_Messages* oMessages = nullptr;
    TsgcOpenAIClass_Run* oRun = nullptr;
    DoLog("[user]: " + memoMessage→Lines→Text);
    Screen→Cursor = crHourGlass; // Change cursor to hourglass
    try
    {
        oMessage = Assistant→CreateMessageText("thread_id", "What is the weather in San Francisco t
        if (oMessage ≠ nullptr)
        {
            oRun = Assistant→CreateRunAndWait("thread_id");
            if (oRun ≠ nullptr)
            {
                oMessages = Assistant→GetMessages("thread_id", oRun→Id);
                if (oMessages ≠ nullptr && oMessages→Messages.Length > 0)
                {
                    memoMessage→Lines→Clear();
                    for (i = 0; i < oMessages→Messages.Length; i++)
                    {
                        DoLog("[assistant]: " + DoFormatResponse(oMessages→Messages[i].ContentText + "\r\n"
                    }
                }
            }
        }
    }
    __finally
    {
        Screen→Cursor = crDefault; // Reset cursor to default
    }
}

```

.NET (C#)

```

public void SendMessage()
{
    DoLog("[user]: " + memoMessage.Text);
    Cursor.Current = Cursors.WaitCursor; // Change cursor to hourglass
    try
    {
        var oMessage = Assistant.CreateMessageText("thread_id", "What is the weather in San Francisco");
        if (oMessage != null)
        {
            var oRun = Assistant.CreateRunAndWait("thread_id");
            if (oRun != null)
            {
                var oMessages = Assistant.GetMessages("thread_id", oRun.Id);
                if (oMessages != null && oMessages.Messages.Length > 0)
                {
                    memoMessage.Text = "";
                    foreach (var message in oMessages.Messages)
                    {
                        DoLog("[assistant]: " + DoFormatResponse(message.ContentText + Environment.NewLine));
                    }
                }
            }
        }
    }
    finally
    {
        Cursor.Current = Cursors.Default; // Reset cursor to default
    }
}

```

6 · Step 2: Upload files and add them to a Vector Store

To access your files, the `file_search` tool uses the Vector Store object. Upload your files and create a Vector Store to contain them.

Delphi (VCL / FireMonkey)

```

procedure UploadFile();
var
  oDialog: TOpenDialog;
begin
  oDialog := TOpenDialog.Create(nil);
  Try
    if oDialog.Execute then
      begin
        Screen.Cursor := crHourGlass;
        Try
          Assistant.UploadVectorStoreFile('sgcVectorStore', oDialog.FileName);
        Finally
          Screen.Cursor := crDefault;
        End;
      end;
  Finally
    oDialog.Free;
  End;
end;

```

C++ Builder

```

void UploadFile()
{
  TOpenDialog* oDialog = new TOpenDialog(nullptr);
  try
  {
    if (oDialog->Execute())
    {
      Screen->Cursor = crHourGlass; // Change cursor to hourglass
      try
      {
        Assistant->UploadVectorStoreFile("sgcVectorStore", oDialog->FileName.c_str());
      }
      __finally
      {
        Screen->Cursor = crDefault; // Reset cursor to default
      }
    }
  }
  __finally
  {
    delete oDialog; // Clean up dialog
  }
}

```

.NET (C#)

```
public void UploadFile()
{
    using (OpenFileDialog oDialog = new OpenFileDialog())
    {
        if (oDialog.ShowDialog() == DialogResult.OK)
        {
            Cursor.Current = Cursors.WaitCursor; // Change cursor to hourglass
            try
            {
                Assistant.UploadVectorStoreFile("sgcVectorStore", oDialog.FileName);
            }
            finally
            {
                Cursor.Current = Cursors.Default; // Reset cursor to default
            }
        }
    }
}
```

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 — OpenAI Chat Completions platform.openai.com/docs/api-reference/chat

Primary standard / spec — Anthropic Messages docs.anthropic.com/en/api/messages

Online help — component page www.esegece.com/help/sgcWebSockets/Components/AI/Applications/ChatBot/TsGcAIOpenAIChatBot.htm

Delphi demo project (in the sgcWebSockets package) `Demos\15.AI\02.Applications\01.ChatBot`

.NET demo project (in the sgcWebSockets package) `.net\demos\15.AI\02.Applications\01.ChatBot`

Component page www.esegece.com/products/websockets/ai/chatbot/

Product page www.esegece.com/products/websockets/

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