API

Message a chatbot

The Chatbot Interaction API allows you to interact with your chatbots using a `POST` request. This API is available for users subscribed to a paid plan and provides a way to communicate with your chatbot programmatically.

Endpoint

```
POST https://www.chatbase.co/api/v1/chat
```

Request Headers

The API request must include the following headers:

- `Authorization: Bearer <Your-Secret-Key>` - The secret key for authenticating the API request.
- `Content-Type: application/json` - The content type of the request payload.

Request Body

The request body should contain the following parameters:

- `messages` (array, required): An array containing the ALL the messages between the user and the assistant. Each message object should have `content` and `role`

properties. The `content` field represents the text content of the message, and the `role` field can be either “user” or “assistant” to indicate the sender of the message.

`chatbotId` (string, required): Refers to the ID of the chatbot you want to interact with (found on the chatbot settings page).

`stream` (boolean, optional, defaults to false): A boolean value indicating whether to stream back partial progress or wait for the full response. If set to `true`, words will be sent back as data-only server-sent events as they become available, with the stream terminated by a `data: [DONE]` message. If set to `false`, the full response will be returned once it’s ready.

`temperature` (number, optional, defaults to 0): What sampling temperature to use, between 0 and 1. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic.

`conversationId` (string, optional): Refers to the ID of the current conversation. The only purpose of this is to save the conversation to the chatbot dashboard. If not provided, the conversation will not be saved. This ID should be generated on your end. You should have different IDs for different conversations and make requests using the same ID for the same conversation. Every time you make a send a message with a specific chatbotId, the last message in the messages array and the response will be added to the conversation.

`model` (string, optional): If this is added to the body, it takes precedence over the model set in the chatbot settings. If not set, the model set in the chatbot settings is used. The option for ‘gpt-4 | gpt-4-turbo’ only works on the Standard and the Unlimited plans.

OpenAI models:

`gpt-4o`

`gpt-4o-mini`

`gpt-4-turbo`

`gpt-4`

`o3-mini`

Claude models:

`claude-3-7-sonnet`

`claude-3-5-sonnet`

`claude-3-opus`

`claude-3-haiku`

Gemini models:

`gemini-1.5-pro`

`gemini-1.5-flash`

`gemini-2.0-flash`

`gemini-2.0-pro`

Cohere models:

`command-r`

`command-r-plus`

DeepSeek models:

`DeepSeek-V3`

`DeepSeek-R1`

Example:

```
{
  "messages": [
    {"content": "How can I help you?", "role": "assistant"},
    {"content": "What is chatbase?", "role": "user"}
  ],
  "chatbotId": "<Your Chatbot ID>",
  "stream": false,
  "temperature": 0,
  "model": "gpt-4o-mini",
  "conversationId": "<Conversation ID generated on your end>"
}
```

Example Request
Chatbase

```
const response = await fetch('https://www.chatbase.co/api/v1/chat', {
  method: 'POST',
  headers: {
    Authorization: 'Bearer <API-Key>',
  },
  body: JSON.stringify({
    messages: [
      {content: 'How can I help you?', role: 'assistant'},
      {content: 'What is chatbase?', role: 'user'},
    ],
    chatbotId: '<Chatbot-ID>',
    stream: false,
    model: 'gpt-4o-mini',
    temperature: 0,
  }),
})

if (!response.ok) {
  const errorData = await response.json()
  throw Error(errorData.message)
}
const data = await response.json()
console.log(data) // { "text": "..."}
```

```
import requests
import json

url = 'https://www.chatbase.co/api/v1/chat'
headers = {
    'Authorization': 'Bearer <API-KEY>',
    'Content-Type': 'application/json'
}
data = {
    "messages": [
        {"content": "How can I help you?", "role": "assistant"},
```

```
        {"content": "What is chatbase?", "role": "user"}
    ],
    "chatbotId": "<Chatbot-ID>",
    "stream": False,
    "temperature": 0
}

response = requests.post(url, headers=headers, data=json.dumps(data))
json_data = response.json()

if response.status_code == 200:
  print("response:", json_data['text'])
else:
  print('Error:' + json_data['message'])
```

```
curl https://www.chatbase.co/api/v1/chat \
  -H 'Authorization: Bearer <Your-Secret-Key>' \
  -d '{
  "messages": [
    {"content": "How can I help you?", "role": "assistant"},
    {"content": "What is chatbase?", "role": "user"}
  ],
  "chatbotId": "<Your Chatbot ID>",
  "stream": false,
  "temperature": 0
}'
```

```
POST /api/v1/chat HTTP/1.1
Host: www.chatbase.co
Authorization: Bearer <Your-Secret-Key>
Content-Type: application/json

{
  "messages": [
```

```
    {"content": "How can I help you?", "role": "assistant"},
    {"content": "What is chatbase?", "role": "user"}
  ],
  "chatbotId": "<Your Chatbot ID>",
  "stream": false,
  "temperature": 0
}
```

Response

The API response will be a JSON object with the following structure:

```
{
  "text": "Chatbase is an AI chatbot builder that lets you create a GPT-based cha
}
```

Example Request with Streaming Functionality

If the `stream` parameter is set to `true`, words will be sent back as data-only server-sent events as they become available. To read the stream, you can use the following code snippets:

```
const response = await fetch('https://www.chatbase.co/api/v1/chat', {
  method: 'POST',
  headers: {
    Authorization: 'Bearer <API-KEY',
  },
  body: JSON.stringify({
    messages: [
      {content: 'How can I help you?', role: 'assistant'},
      {content: 'What is chatbase?', role: 'user'},
    ],
    chatbotId: '<Chatbot-ID>',
```

```
    stream: true,
    temperature: 0,
    model: 'gpt-4o-mini',
  }),
})

if (!response.ok) {
  const errorData = await response.json()
  throw Error(errorData.message)
}

const data = response.body

if (!data) {
  // error happened
}

const reader = data.getReader()
const decoder = new TextDecoder()
let done = false

while (!done) {
  const {value, done: doneReading} = await reader.read()
  done = doneReading
  const chunkValue = decoder.decode(value)
  console.log(chunkValue) // This will log chunks of the chatbot reply until the
}
```

```
import requests
import json

url = 'https://www.chatbase.co/api/v1/chat'
headers = {
    'Authorization': 'Bearer <API-KEY>',
    'Content-Type': 'application/json'
}
data = {
    "messages": [
```

```
        {"content": "How can I help you?", "role": "assistant"},
        {"content": "What is chatbase?", "role": "user"}
    ],
    "chatbotId": "<Chatbot-ID>",
    "stream": True,
    "temperature": 0
}

response = requests.post(url, headers=headers, data=json.dumps(data), stream=True

if response.status_code != 200:
    json_data = response.json()
    print('Error:' + json_data['message'])

else:
  decoder = response.iter_content(chunk_size=None)
  for chunk in decoder:
    chunk_value = chunk.decode('utf-8')
    print(chunk_value, end='', flush=True)
```

Saving Conversations

In order to save conversations to the dashboard, a `conversationId` parameter needs to be included in the request body. The full conversation needs to be sent on every API call because Chatbase doesn't save previous messages. The messages received in the latest API call for a given `conversationId` overrides the conversation already saved there.

Error Handling

If there are any errors during the API request, appropriate HTTP status codes will be returned along with error messages in the response body. Make sure to handle these errors gracefully in your application.

That's it! You should now be able to message a chatbot using the Message API.