API Documentation

Commented [PA1]: @Rohit Poduval (ropoduva) @Siva Kurapati (sivakura) I have formatted the document, see if it is all good, and we can share this documentation.

Commented [S(2R1]: everything looks good. made small changes. we should find a way to include notebooks sample notebooks as well.

Table of Contents

API Documentation......1

Chat Completion API......2

Overview......2

API Endpoint......2

Authentication......3

Obtaining an Access Token......3

Request......4

Response......5

Using OpenAI package:......5

Jupyter Notebook with above details:......6

Chat Completion API

Overview
This document provides information on how to use the CircuIT API provided by Cisco. The API allows you to interact with the **OpenAI** models for generating chat completions.

To use this API, you will need the following:

- Okta credentials (clientid and client secret) for authentication.
- An 'appkey' for identifying your application.

If you do not have the required credentials or appkey, please request one using the API request form.

API Endpoint

- **Endpoint URL:** `https://chat-ai.cisco.com/openai/deployments/<model name>/chat/completions`

Supported Models & API Versions

Model Name	**API Version**	**Context Windows**	**Available in Free Tier (Restrictions apply)**
gpt-4.1	**2025-01-01-preview**	**120K Tokens (Free Tier)** **1M Tokens – Pay-as-you-use tier**	**Yes**
gpt-4o-mini	**2025-01-01-preview**	**120K Tokens**	**Yes**
gpt-4o	**2025-01-01-preview**	**120K Tokens**	**Yes**
o4-mini	**2025-01-01-preview**	**200k Tokens**	**No**
o3	**2025-01-01-preview**	**200k Tokens**	**No**
gemini-2.5-flash	**2025-01-01-preview**	**1M Tokens**	**No**
gemini-2.5-pro	**2025-01-01-preview**	**1M Tokens**	**No**

Deprecated Models that are no longer available

- **gpt-4**
- **gpt-35-turbo**
- **gpt-35-turbo-16k**

Authentication

The API requires authentication using an access token obtained via the OAuth2 authentication flow using your Okta credentials (clientid and client secret).

Obtaining an Access Token (used as api-key)

To obtain an access token, you can use the following cURL command:

```
client_id=your_client_id
client_secret=your_client_secret

# Base64 encode the client_id and client_secret
encoded_value=$(echo -n "${client_id}:${client_secret}" | base64)

# Run the curl command
curl --location --request POST
'https://id.cisco.com/oauth2/default/v1/token' \
--header 'Accept: */*' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--header "Authorization: Basic ${encoded_value}" \
--data-urlencode 'grant_type=client_credentials'-dw
```

To generate the `<base64_encoded_value>` for the 'Authorization' header, you can use the following command:

```
echo -n <client_id>:<client_secret> | base64
```

Note #1: For your `clientid` and `clientsecret – if you had requested API access, this would likely have been shared with you as an information card.`

Note #2: When this text is copied from Word, the editor often inserts new lines and spaces following backslashes (\). To prevent errors with the curl command, ensure these are removed.

NOTE #3: Access token expiry - that the access token expires every hour and needs to be re-generated when it expires.

Sample Python Code to Generate the Access Token

```
import requests, json
import base64

url = https://id.cisco.com/oauth2/default/v1/token

payload = "grant_type=client_credentials"
value = base64.b64encode(f'{client_id}:{client_secret}'.encode('utf-
8')).decode('utf-8')
headers = {
    "Accept": "*/*",
    "Content-Type": "application/x-www-form-urlencoded",
    "Authorization": f"Basic {value}"
}
token_response = requests.request("POST", url, headers=headers,
data=payload)
token_data = token_response.json()
api_key = token_data.get('access_token')
```

Request

Sample cURL Request

```
curl --location 'https://chat-ai.cisco.com/openai/deployments/gpt-4o-
mini/chat/completions' \
--header 'Content-Type: application/json' \
--header 'Accept: application/json' \
--header 'api-key: <access_token>' \ # use access_token from above
--data '{
  "messages": [
{
"role": "system",
"content": "You are a chatbot"
},
    {
      "role": "user",
      "content": "who is the president of USA."
    }
  ],
  "user": "{\"appkey\": \"<appkey>\"}", #Please reach out for appkey to be used
  "stop": ["<|im_end|>"]
}'
```

Request Parameters

Parameter	Type	Description
`messages`	Array	An array of message objects.
`user`	string	A JSON string containing the `appkey` information.
`stop`	Array	An array used for stopping the chat completion. Leave it empty (`[""]`) for continuous conversation.
`api-key`	Header	Your access token obtained through OAuth2 authentication.

messages Array

- The `messages` array contains message objects.
- Each message object has a `role` (either "user" or "assistant") and `content` (the content of the message).

`user` JSON Object

- The `user` object should contain your `appkey`, `session_id`, and `user` information.
- The `appkey` is a required field to identify your application.
- `session_id` – Optional parameter (include if you want to maintain conversational history)
- `user` – Optional parameter (cec id). Used to identify the user making the request

Response
The API response will contain the chat completion generated by the GPT-3.5 Turbo model.

That's the API documentation for interacting with the Chat AI API provided by Cisco. Make sure to replace placeholders like `<access_token>` and `<appkey>` with your actual values when making API requests. If you have any questions or require additional assistance, please feel free to contact the Chat AI API Webex Space .

Using OpenAI package (>1.0.0):
Below is the sample using openai python package.

```
# !pip install openai
import os
from openai import AzureOpenAI


client = AzureOpenAI(
  azure_endpoint = 'https://chat-ai.cisco.com',
  api_key=token_response.json()["access_token"],
  api_version="2024-08-01-preview"
)


response = client.chat.completions.create(
    model="gpt-4o-mini", # model = "deployment_name".
    messages=message_with_history,
    user=f'{{"appkey": "{app_key}"}}'
)
print(response.choices[0].message.content)
```

Sample Jupyter Notebooks: Notebooks
Please use the Webex Space for help if you face issues with API access/usage