Paul Borisov

Azure OpenAI Chat Web Part, v1.3.4

Supported options: chat history, private and global chat sharing, consecutive event streaming, integration with Azure API Management, options to use encrypted SharePoint list / Local storage / Azure SQL Database for storing chat history, options to support full-screen mode and unlimited length of chat history, integrations with SharePoint search, company users, local date and time, search on Internet using Bing and Google services, image generation with Dalle3, image analysis with GPT-4o, PDF content analysis, voice input and read out of AI-generated text.

Latest revision from 4-18-2025

Table of Contents

Introduction

This document outlines the functionality of the Azure OpenAI Chat Web Part, its frontend and backend components, and configurations.

It refers to the supplementary document, **azure-openai-chat-security.pdf**, which offers more detailed information on configurable security options.

Deployment

To deploy the web part, you should add its compiled solution, **azure-openai-chat.sppkg**, into an App Catalog.

- This can be a global App Catalog of your SharePoint Online tenant, or a site collection scoped one.
- Use standard deployment options to make the web part globally available on all sites or add the app manually to specific sites.

Do you trust Azure OpenAI Chat Web Part?

The client-side solution you are about to deploy contains full trust client side code. The components in the solution can, and usually do, run in full trust, and no resource usage restrictions are placed on them.

This client side solution will get content from the following domains:

SharePoint Online

☐ Make this solution available to all sites in the organization

Please go to the API Management Page to approve pending permissions. These are the permissions that need to be reviewed: openaiwp, user_impersonation; Microsoft Graph, People.Read; Microsoft Graph, User.Read.All

Permissions

After adding the package, open the URL https://**yourtenant**-admin.sharepoint.com/_layouts/15/online/AdminHome.aspx#/webApiPermissionManagement and approve the requested permissions.

To enable this permission, create a standard Azure App registration named **openaiwp** in Microsoft Entra ID (Azure AD). The web part uses this App registration to make authenticated requests using the context of the signed-in user.

This is used in the Chat sharing option. It is utilized to get details of all users from AAD to display them in the People Picker component.

An alternative (reduced) permission for the Chat sharing option is available. If access level User.Read.All cannot be granted, the web part can use this reduced scope to get only the basic information of the user's colleagues. In this case, the list of available contacts may be reduced compared to the more advanced previous option. If neither of the last two permissions can be granted, the Chat sharing feature can only provide unrestricted sharing.

| Permission	Scope	Purpose
openaiwp, required	user_impersonation	To enable this permission, create a standard Azure App registration named **openaiwp** in Microsoft Entra ID (Azure AD). The web part uses this App registration to make authenticated requests using the context of the signed-in user.
Microsoft Graph, optional	User.Read.All	This is used in the Chat sharing option. It is utilized to get details of all users from AAD to display them in the People Picker component.
Microsoft Graph, optional	People.Read	An alternative (reduced) permission for the Chat sharing option is available. If access level **User.Read.All** cannot be granted, the web part can use this reduced scope to get only the basic information of the user's colleagues. In this case, the list of available contacts may be reduced compared to the more advanced previous option. If neither of the last two permissions can be granted, the Chat sharing feature can only provide unrestricted sharing.

Configuration

Create a modern Site Page, add the web part "Azure OpenAI Chat", and click on the "Configure" button.
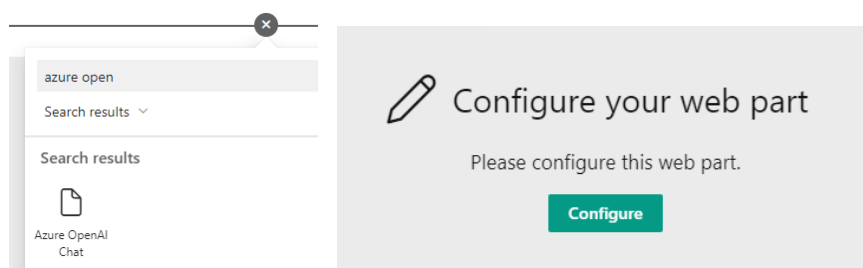
This action opens the web part settings with the default values shown below.

Azure OpenAI Chat   ✕

Settings

Client ID: create a user_impersonation app with name=openaiwp

```
00000000-0000-0000-0000-000000000000
```

Base URL for GPT endpoint (APIM API or full)

```
https://tenant.azure-api.net/openai
```

Base URL for GPT4 endpoint (APIM API or full)

```
https://tenant.azure-api.net/openai4
```

Base URL for Chat WebApi (APIM API or full)

```

```

Language models

- ☑ GPT-3.5
- ☑ GPT-4
- ☐ GPT-4 Turbo (beta, not for Production)
- ☐ GPT 4o
- ☐ GPT 4o Mini

Storage type for chat history

```
SharePoint list                          ⌄
```

SharePoint list URL (leave it empty for default URL)

```

```

Update

- ☐ Enable storage encryption
- ☐ Enable sharing
- ☑ Enable streaming
- ☑ Enable full screen mode
- ☑ Enable integrations
- ☑ Bing search

Optional api-key for Bing

```
Add if APIM endpoint is not configured
```

➕

Azure OpenAI Chat   ✕

SharePoint list URL (leave it empty for default URL)

```

```

Update

- ☐ Enable storage encryption
- ☐ Enable sharing
- ☑ Enable streaming
- ☑ Enable full screen mode
- ☑ Enable integrations
- ☑ Bing search

Optional api-key for Bing

```
Add if APIM endpoint is not configured
```

- ☑ Google search

Optional key for Google: key=API_KEY&cx=SEARCH_ENGINE_ID

```
Add if APIM endpoint is not configured
```

- ☑ Image analysis: configure /vision (docs)
- ☑ Image generation: configure /dalle (docs)

Image library URL (leave it empty for default URL)

```

```

Update

- ☑ Enable examples for the prompt text
- ☑ Enable voice input
- ☑ Enable voice output (text to speech)
- ☑ Code highlighting
- ☐ Show highlighting styles

Default style

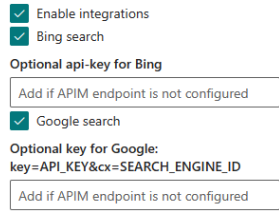```
vs2015   ⌄
```

- ☐ Show prompt area at bottom
- ☑ Unlimited chat history length (AI-responses in long chats may be less accurate)

**Locale for dates (default is fi-FI)**

```

```

| Title | Description | Default |
|---|---|---|
| Client ID: create a user_impersonation app with name=openaiwp | The App ID (Client ID) for the **openaiwp** Azure registration is used to authorize SPFx requests. | 00000000-0000-0000-0000-000000000000 |
| Base URL for GPT endpoint (APIM API or full) | This is the endpoint for the primary language model "GPT 3.5".<br><br>You should either deploy the endpoint via the Azure API Management service (recommended for greater security) or use the full URL to an Azure OpenAI endpoint (https://tenant.openai.azure.com/...) with the api-key added to its own setting below. | https://tenant.azure-api.net/openai<br><br>This refers to the endpoint published via the Azure API Management service. |
| Base URL for GPT4 endpoint (APIM API or full) | This is an optional endpoint for the language model "GPT 4".<br><br>The same rules as above apply. | https://tenant.azure-api.net/openai4<br><br>Same as above. |
| Base URL for Chat WebApi (APIM API or full) | This is an optional endpoint for the "ChatWebAPI" App Service, which manages chat history data in the Azure SQL database.<br><br>You should either deploy the endpoint via the Azure API Management service (the recommended secure method) or use full URL to your version of **ChatWebAPI**. | Empty (it defaults to https://tenant.azure-api.net/chatwapi)<br><br>Same as above. |
| Optional api-key for Azure OpenAI (for troubleshooting, not for Production) | Please note that if you use your api-key here, it poses a security risk as the key could potentially be leaked, because it will become visible in the browser's request headers.<br><br>The key is encrypted and stored in the web part settings (displayed as ***). |  |
| Language models | There are several predefined language models, GPT 3.5 (gpt-35-turbo-16k) , GPT 4 (gpt-4-32k), and GPT 4 Turbo (gpt-4-1106-preview. They can become outdated while the new models have been added.<br><br>v1.3 (September 2024): support of two newer models was added (webpart version 1.3+)<br>• GPT-4o | GPT 3.5, GPT 4, GPT-4 Turbo, GPT-4o, GPT-4o Mini, GPT-4.1, GPT-4.1 Mini, GPT-4.1 Nano, O1, O1 Mini, O3 Mini, O4 Mini.<br><br>If only one language model is toggled, the language model selector will not be visible in the UI. If none is selected, the |

| | | |
|---|---|---|
| | • GPT-4o Mini<br><br>v1.3.3 - 1.3.4 (April 2025): support for new models was added:<br>• gpt-4.1<br>• gpt-4.1-mini<br>• gpt-4.1-nano<br>• o1<br>• o1-mini<br>• o3-mini<br>• o4-mini<br><br>You should use predefined deployments in APIM-endpoints /openai and /openai4 published via API Management.<br><br>Alternatively, if you enter full URLs to Azure OpenAI endpoints, a textbox with model names appears instead of predefined checkboxes. In this box, you can adjust the model names as needed.<br><br>Be careful with the adjustments if you use full URLs to Azure OpenAI endpoints.<br>• The logic uses exact model names to dynamically modify the endpoint URLs when it is necessary. For example, it can dynamically change /openai/deployments/**gpt-4-32k**/ to /openai/deployments/**gpt-4-0613**/chat/completions at the time of request execution.<br>• APIM-based endpoints are predefined. They use the deployments defined in APIM operations. They do not use any dynamic adjustments. | logic defaults to using the gpt-35-turbo model. |
| Storage type for chat history | There are three predefined options for storage: SharePoint list, Local Storage, and (Azure SQL) Database.<br><br>The default option stores chat history in a SharePoint list. When you select this, a textbox and a button become available to create a custom list. By default, the list is created on the current site with the predefined name, dbChats. Security settings | SharePoint list<br><br>The option for SharePoint list supports chat privacy. For example, SharePoint search does not display private chat conversations to users other than the authors of those chats. If users open the Custom list **dbChats**, they will |

| | | |
|---|---|---|
| | are automatically enabled for list items to limit access to their authors. The SharePoint Search Crawler does not index the list content by default to prevent potential leaks of private conversations.<br><br>The option of retrieving/saving chat history to/from an Azure SQL database is managed by the **ChatWebAPI** App Service (as mentioned above).<br><br>You can use "Local Storage" to quickly demonstrate the functionality to customers. In this case, the chat history will be saved in the local storage of your browser. While actual chat sharing will not work, you can demonstrate this option too. | only see their own messages. However, this does not apply to Site owners, who can see all records in the list and can find them in SharePoint search results.<br><br>Please note if you enable the sharing feature while using SharePoint list storage, you should adjust the list permissions using the Update button.<br><br>In case of using the SharePoint list storage, the maximum length of Chat history text is limited to 2 MB per Chat entry. This is the technical limit of the multiline text field in SharePoint Online. |
| Enable storage encryption | `Enables AES encryption of the sensitive personal data in Chats for the selected storage type:`<br>• SharePoint list<br>• Local storage<br>• Database<br><br>Encryption transparently supports Shared chats, private and global.<br><br>Using encryption is a reasonable choice when you use SharePoint list storage with the sharing option enabled and want to hide details of personal Chats from other users of SharePoint. | Not selected<br><br>Sensitive personal data includes Chat name and Chat messages. When the storage encryption is enabled, the display name of the current user is not saved (set to empty). |
| Enable sharing | Enables a sharing option for personal Chats. This option is disabled by default.<br><br>Chat authors can decide if they want to share their personal Chats with others. | Selected<br><br>The web part supports two levels of sharing:<br>1. Global sharing: each personal Chat can be shared with Everyone.<br>2. Private sharing: each personal Chat can be shared with up to 15 specific users selected in the dialog. |

| | | |
|---|---|---|
| Enable streaming | Enables the event-streaming response option, which provides gradual outputs of texts generated by AI. This option is disabled by default. | Selected |
| Enable full screen mode | Enables full-screen mode and displays the expansion icon ⤢ in the upper right-hand corner. | Selected |
| Enable integrations | Enables integrations with external data using the feature of OpenAI "Function calling". There are several functions available by default in the web part:<br>• searchSharepoint<br>• peopleSearch<br>• currentDateAndTime<br><br>There are also optional functions configured separately if corresponding services are available:<br>• searchOnInternet: this function uses Azure Bing Search service with api-key managed by APIM or configured explicitly in the web part settings (encrypted).<br>• searchOnGoogle: this function uses Google Custom Search with key=…&cx=… values managed by APIM or configured explicitly in the web part settings (encrypted). | Not selected<br><br>☑ Enable integrations<br>☑ Bing search<br>**Optional api-key for Bing**<br>[ Add if APIM endpoint is not configured ]<br>☑ Google search<br>**Optional key for Google:**<br>**key=API_KEY&cx=SEARCH_ENGINE_ID**<br>[ Add if APIM endpoint is not configured ]<br><br>Configuration instructions for optional Bing and Google APIM endpoints are described in the chapter **Azure API Management** below. |
| Image analysis (Vision) | Enables the option to explore and describe uploaded images using GPT-4 Vision model of (Azure) OpenAI.<br><br>The APIM endpoint /vision must be configured to use this option. Configuration instructions for adding this endpoint are described in the chapter **Azure API Management** below.<br><br>Alternatively, if you already use the Native OpenAI endpoint URL for GPT 4 chats, it will automatically use the corresponding endpoint to analyze images using the same **api-key** stored in the web part settings (encrypted).<br>• Native OpenAI uses the endpoint https://api.openai.com/v1/chat | Not selected |

| | | |
|---|---|---|
| | /completions with gpt-4-vision-preview model<br><br>Also, you can use Azure OpenAI service URL with the predefined endpoint https://tenant.openai.azure.com /openai/deployments/gpt-4-vision-preview/images /generations?api-version=2023-07-01-preview available in Swedish Central zone (with Computer Vision resource deployed into the same zone) | |
| Image generation (Dalle) | Enables the option to generate images using Dalle3 imaging model of (Azure) OpenAI.<br><br>The APIM endpoint /dalle must be configured to use this option. Configuration instructions for adding this endpoint are described in the chapter **Azure API Management** below.<br><br>Alternatively, if you already use the full Azure OpenAI or Native OpenAI endpoint URL for GPT 3.5 chats, it will automatically use the corresponding endpoint to generate Dalle3 images using the same **api-key** stored in the web part settings (encrypted).<br> • Azure OpenAI service uses the endpoint https://tenant.openai.azure.com /openai/deployments/dalle3/images /generations?api-version=2023-12-01-preview available in Swedish Central zone<br> • Native OpenAI uses the endpoint https://api.openai.com /v1/images/generations | Not selected<br><br>In Azure OpenAI, the deployment of the model Dalle3 is available in Swedish Central zone (as of December 2023).<br><br>Native OpenAI provides the model Dall-e-3 available by default in paid API subscriptions.<br><br>If you enable the option for Image generations, you must create the SharePoint library, which will store generated images. Just click on the "Create" button below the checkbox to created it automatically.<br><br>☑ Image generation (Dalle)<br><br>Image library URL (leave it empty for default URL)<br><br>[          ]<br><br>**Create**<br><br> • Default library is created on the current site with the name **Chat Images**.<br> • Generated images are excluded from SharePoint Search results. However, they are available for |

| | | SharePoint users if they access the library content directly.<br><br>As of December 2023, the web part also includes the older processing logic for the previous Dalle2 imaging model kept for backward compatibility. |
|---|---|---|
| Enable examples for the prompt text | Enabled the dropdown box visible in the left-hand side of the prompt area. It opens the list of available examples for the prompt text that include integration samples (when enabled). | Not selected |
| Enable voice input | If the browser supports Web Speech API and the microphone is available, this option provides voice input for the prompt text. | There are 15 popular world languages available by default. |
| Enable voice output (text to speech) | If the browser supports Web Speech API, this option provides reading out of AI-generated texts on click of the Radio-icon.<br><br>Alternatively, if you use Native OpenAI endpoints, this option supports its tts-1-hd model (text to speech). In this case the language to read out texts is selected automatically by OpenAI. | There are 15 popular world languages to read out texts in available by default. |
| Code highlighting | Enables code highlighting to render markdown outputs of code snippets. | Selected |
| Show highlighting styles | Enables the display of a dropdown with code styles for highlighted outputs. This feature is disabled by default. | Selected |
| Default style | Default code style for highlighted outputs. | stackoverflowDark |
| Show prompt area at bottom | When enabled, the prompt text area will appear at the bottom. It is disabled by default. | Not selected |
| Unlimited chat history length (AI-responses in long chats may be less accurate) | When enabled, this feature permits the use of unlimited history length in chats. In such cases, the system dynamically removes earlier messages just prior to submitting the history to Azure OpenAI. This process does not impact the storage of the entire (uncut) history and new responses. | Not selected |
| Locale for dates (default is fi-FI) | The locale for formatting dates visible in the web part is optional. The default setting is | Default (empty textbox) |

| | 'fi-FI' (empty), but it can be changed, for instance, to 'en-US'. | |
|---|---|---|

## Privacy settings for SharePoint List storage

If you plan to create a SharePoint list to store chat histories, ensure that this does not expose data to the Search Crawler. By default, the list creation button applies the following changes to list settings:

- Permissions for this list: Unique (permission inheritance is broken)
    - Everyone except external users: Contribute
    - List's creator: Full Control
- Advanced settings: Read and write permissions are only for item authors
    - Read access: Read items that were created by the user.
      **Important:** If you wish to use shared chats, you will need to adjust access to less strict value "Read access: Read all items". In this context, users may gain access to others' messages. This is a technical limitation of SharePoint storage.
        - To address this issue, you can enable storage encryption.
        - To adjust permissions automatically, click on the Update button for the SharePoint list in the web part settings.
    - Create and Edit access:  Create items and edit items that were created by the user
- Allow items from this list to appear in search results? No
    - Note: You can modify this setting if necessary. In this case list items should be accessible in search results for their authors and site admins. However, they should not be available to regular users unless you use Read access: Read all items as mentioned above.

## Privacy settings for generated images

AI-generated images are not private. They are stored in the document library available for Everyone by default.

If you enable the option for Image generations, the corresponding library **Chat Images** uses the following default settings:

- Permissions for this list: Unique (permission inheritance is broken)
    - Everyone except external users: Contribute.
        - You can create your own Custom permission level, for example, Contribute without Delete.
    - List's creator: Full Control
- Allow items from this list to appear in search results? No
    - You can modify this setting if necessary.

## Storage encryption

When you enable storage encryption, the maximum chat name length is limited to 150 chars. Unencrypted storage supports up to 255 characters.

## Options to use native Open AI instead of Azure OpenAI

The web part also supports the use of native OpenAI API URLs, such as https://api.openai.com/v1/chat/completions, instead of Azure OpenAI ones.

You can use the native OpenAI API deployed via the API Management service with predefined endpoint(s) that contain the word "**native**".

- For example, you might use https://tenant.azure-api.net/openainative or https://tenant.azure-api.net/openainative4 or both.
- In such cases, the logic will correctly manage conditions for querying data from the native OpenAI API, including automatic adjustments of model names - like gpt-3.5-… instead of gpt-35-…, smoother event streaming outputs, etc.

To conduct quick tests, you can temporarily use the direct URL https://api.openai.com/v1/chat/completions in the web part settings for "Base URL for GPT endpoint (APIM API or full)" and "Base URL for GPT endpoint (APIM API or full)". You can add the api-key for native Open AI into the setting "Optional api-key for Azure OpenAI (for troubleshooting, not for Production)". For security reasons, it is recommended to use API Management service deployments, as described above.
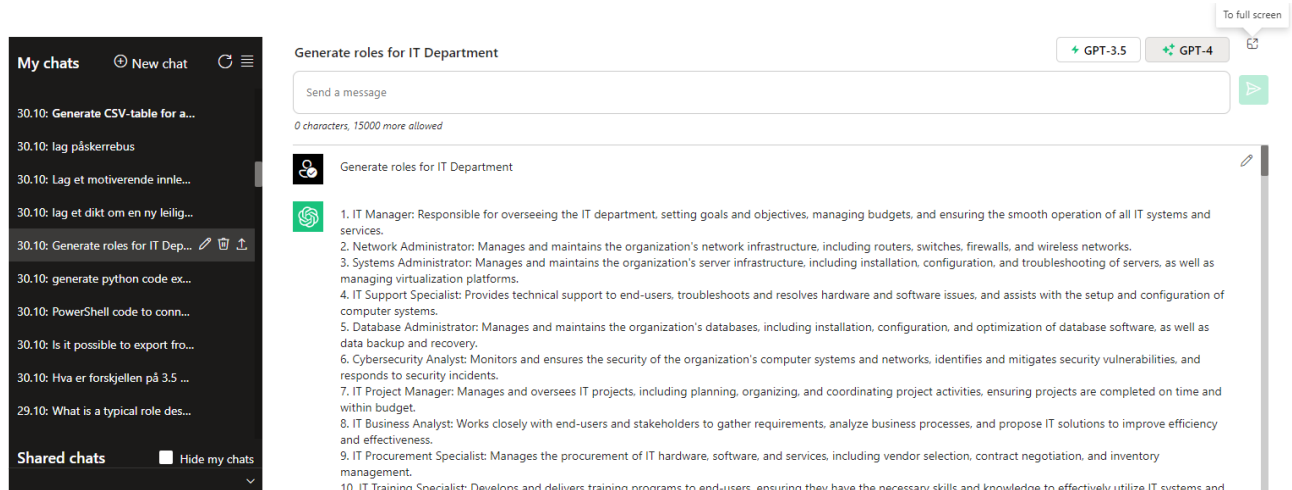
## User interface

**Global version**

This is the initial UI without chat history.



This is the UI displaying the previous chat history of the current user.

## Navigation panel

The left-hand panel displays chats started by the current user and sorted by the dates of the last modifications.

- When the user starts a new chat, the logic automatically generates a chat name using the entered texts.
- The default date format corresponds to Finnish: dd.MM for the current year and dd.MM.yyyy for previous years. This can be changed in the web part settings, for example, to en-US.
- The user can edit the chat name, delete the chat, and share it with others (if the sharing option has been enabled in the settings). The actions of deleting and sharing require the user's confirmation.
- Corresponding action icons appear after the chat name when the user selects the chat's row.
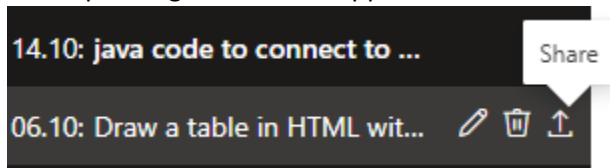


- The sharing dialog offers two options: "Share with Everyone" (which is the default) and "Share with specific users." In the latter case, only the selected users will have access to the chats shared with them.
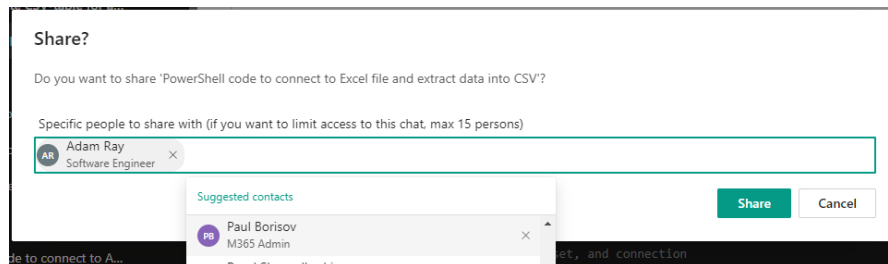


- Shared chats accessible to the current user are displayed at the bottom of the Navigation Panel. Users can select the "Hide My Chats" checkbox if they wish to hide the chats they have shared.
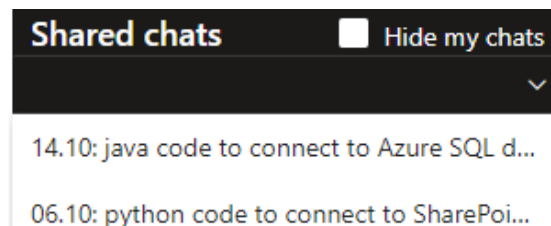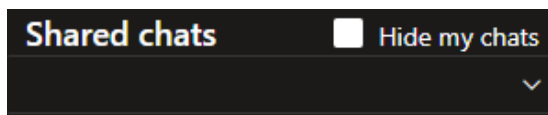


- The web part supports code highlighting options, which are accessible in the settings. If "Show Highlighting Styles" is enabled, the corresponding selector will appear at the bottom.
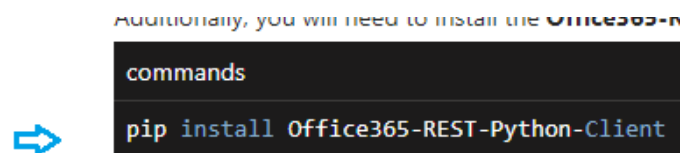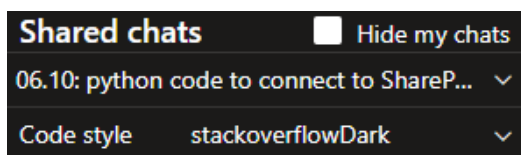
## Content panel

This area displays the message history for the chat selected in the Navigation Panel. By default, the most recently modified chat by the user is selected.

- The upper section of the panel displays the chat's name and, if more than one model is available in the settings, the language models. An icon to switch to Full Screen mode is also present, providing convenience for longer chats.
- A prompt area is designated for entering questions, followed by a character counter. The maximum request length is restricted to 15,000 characters for GPT-3.5-turbo-16k 30,000 characters for GPT-4-32k and 125,000 characters for GPT-4-1106-preview, GPT-4o and GPT-4o Mini. Note that newer models like gpt-4.1 support 10x context window (1mln+ characters).
- The maximum acceptable history length for a single chat should not exceed 64k characters for GPT-3.5-turbo-16k and 128k characters for GPT-4-32k. If this length is exceeded, the user should initiate a new chat to continue the conversation.



  - There is an option in the web part settings for "Unlimited Chat History Length (AI-responses in long chats may be less accurate)". When this feature is enabled, it allows for an unlimited history length. In such cases, th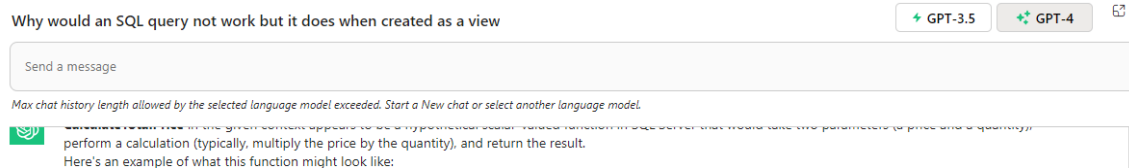e system automatically removes the earliest messages before submitting the history to Azure OpenAI. However, this does not impact on the saving of the entire (uncut) history and new responses back into storage.

- Chat messages are composed of user queries followed by AI responses. If a user makes a typo, they can edit the message, correct it, and save the changes. Subsequent queries will use this updated history.
- If AI responses include code snippets, these are displayed in a highlighted format. The user can:
  - Click on the 'Raw' button ( ☰ ) to view the original, unformatted response.
  - Click on the 'Copy' button ( 🗍 ) to copy the code snippet to the clipboard.

To full screen

java code to connect to Azure SQL database    ⚡ GPT-3.5    ✦ GPT-4

Send a message

0 characters, 7000 more allowed

java code to connect to Azure SQL database

Sure! Here's an example of Java code to connect to an Azure SQL database using the JDBC driver:

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class AzureSQLConnectionExample {
    public static void main(String[] args) {
        // Azure SQL database connection details
        String server = "your_server.database.windows.net";
        String database = "your_database";
        String user = "your_username";
        String password = "your_password";
        String url = String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;"
                + "hostNameInCertificate=*.database.windows.net;loginTimeout=30;", server, database, user, password);
```

Using Shared chats

When users select a Shared Chat at the bottom of the Navigation Panel, they initiate a new chat, and a copy of the chat's history is loaded into the Content Panel. Subsequently, users can continue the original conversation within their own copies of the Shared Chat. For security reasons, it is not permitted to continue original conversations on behalf of their authors.

Localization

The web part uses standard localization files for SPFx solutions. The current version includes files for the following languages:

- English (en-us)
- Finnish (fi-fi)
- Norwegian (nb-no)

Event streaming

If the streaming option is enabled in the web part settings, the AI responses should be presented with the visual effects of consecutive text outputs, with the entire response being formatted at the end of the process.

java code to connect to Azure SQL database

Sure! Here's an example Java code snippet to connect to an Azure SQL database using JDBC:

```
```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class AzureSQLConnection {
    public static void mai ...
```

If the streaming option is not enabled, the complete, formatted AI response will be provided at the end of the processing stage.

Voice input

This option was added in December 2023. It uses Web Speech API if it is supported by the browser. There are 15 popular world languages available by default. Voice input was tested for all of them.

To use the voice input, click on the MIC-icon near the submit button, select the desired language, and start speaking into your microphone. Click on the stop button to input your speech into the prompt box and then click on the Submit button as usual.
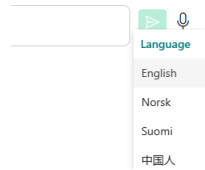
Voice output

This feature is supported in two options.

Web Speech API

Most modern browsers support Web Speech API. It provides a straightforward way to use speech synthesis for AI-generated texts when OpenAI text-to-speech model is not available.

- Click on the Radio-icon in the right-hand part of the content area.
- The standard Web Speech API requires selecting the preferred language; using default page language is not always optimal. There us a dropdown menu to select the language to read out texts. The read out starts upon the selection.
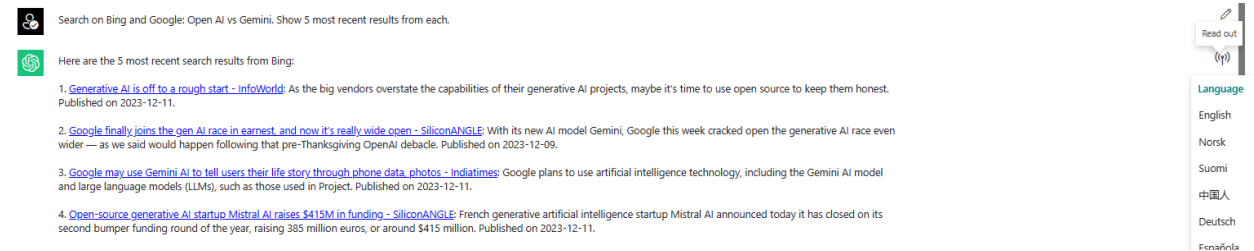
Native OpenAI

Native OpenAI has a separate language model tts-1 (tts-1-…) that handles text to speech conversions. This model provides automatic handling of texts that contain mixed languages.

Azure OpenAI does not yet have support for text-to-speech models (as of Dec 2023).

If you use any direct Native OpenAI endpoint in the web part settings, tts-1 is supported automatically. Alternatively, if you use the APIM endpoint /openainative, you can configure its /tts operation (/openainative/tts) to support text to speech conversions. You can find configuration details in the Backend chapter below.

Examples for prompt text

Click on the dropdown menu and select an example. It will be inserted into the prompt area.

| New chat | | ⚡ GPT-3.5 | ✦ GPT-4 | ✦ GPT-4 Turbo | ⬆ | ⛶ |

Examples

Create an image: an illustration of a business meeting

Search on the Internet: Sam Altman's ouster

Search on Bing: Microsoft Copilot, most recent news

Search on Google: Gemini AI, most recent news. Add Reddit.

Search on Bing and Google: Open AI vs Gemini. Show 5 most recent results from each.

Search on SharePoint: Resource Management System. Format the results as an HTML table.

People search: all users with the name John. Roles and emails.

Get company users that have names starting with K. Format the results as an HTML table.

The current date and time

Generate CSV-table for all weeks of 2023. Columns: WeekNumber, StartDate:, EndDate. Use semicolon as column delimiter and ISO-date format. Start day is Monday. End day is Friday. Wrap into ```

Powershell code to connect to Azure SQL database

Image analysis (Vision)

When this optional integration is enabled, you can select an image or images using the upward arrow button located in the right-hand corner.

- After selecting the images, the prompt texts "What is in these images" is added automatically.
- Click on the Submit button to make their analysis.
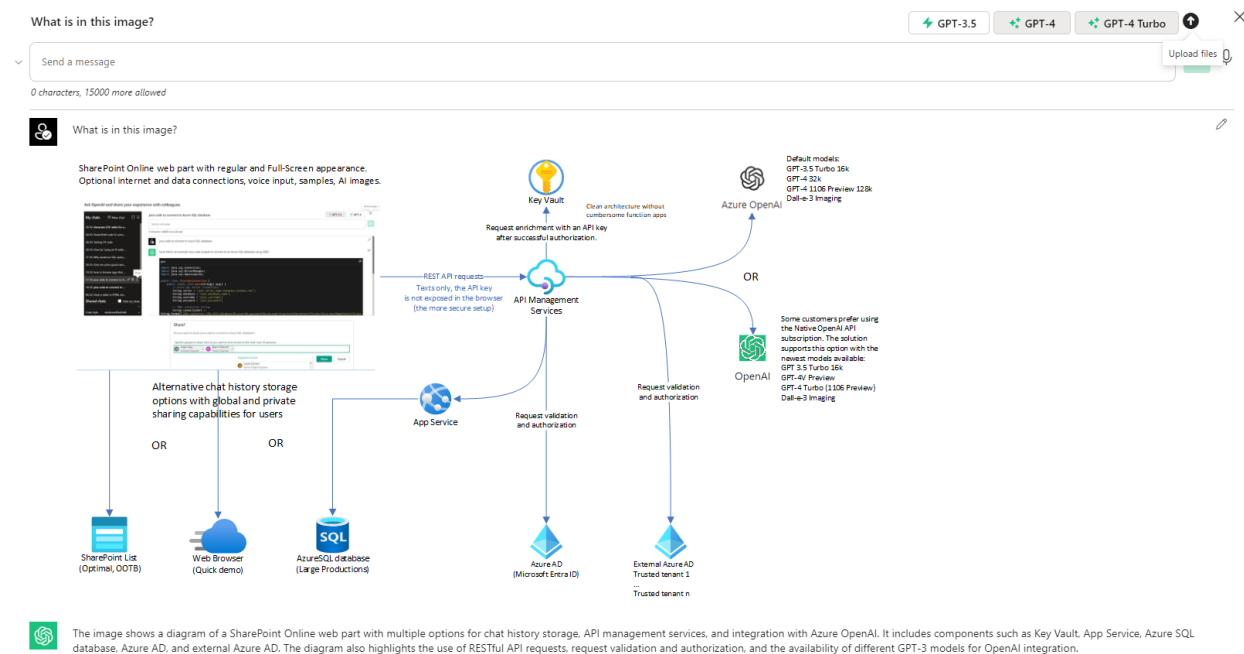
Image generations (Dalle)

When this optional integration is enabled, you can use the prompt text like shown below to generate an image

- Create an image: an illustration of a business meeting
- Click on the Submit button to generate the image.

Illustration of a business meeting where a diverse group of four individuals are seated at a table. A Caucasian woman is presenting her ideas on a chart, an Asian man is taking notes on his laptop, an African-American woman is listening attentively, and a Hispanic man is brainstorming along. Each participant exhibits professional attire, a reflective atmosphere fills the room with ideas and discussions.

Backend

Azure OpenAI service

As of November 2023, the options to create this service are not readily available in the Azure portal. Each new customer is required to complete a specific electronic form with their company details, submit it to Microsoft, and await approval for the service. Typically, Microsoft reviews and approves the request within a few working days.

The Azure OpenAI service can be created in any zone that supports GPT 3.5 and GPT 4 deployments. For instance, Swedish Central and East US 2 are suitable choices as they support both models.

- Please note that European zones may support a limited range of text models. Usually, the latest models are first introduced in US zones before they become available in EU zones.
- As of December 2023, Dalle3 imaging model is only available in Swedish Central. Consider choosing this zone if you wish to use it for Image generations in the web part.

Model deployments

Add two models to Azure OpenAI service instance using the blade Resource Management > Model deployments > Manage deployments. Configure the predefined deployment names as shown in **bold**.

- **gpt-35-turbo-16k**, version 0613 (Dec 2023). According to Microsoft, it uses training data up to Sep 2021.
- **gpt-4-32k**, version 0613 (Dec 2023). According to Microsoft, it uses training data up to Sep 2021.
- You can use other models as well: https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/models#model-summary-table-and-region-availability

Optionally, it is recommended to add the deployment for **gpt-4-1106-preview** (GPT 4 Turbo), which supports 128k input tokens and uses training data up to April 2023.

- As of December 2023, this model is not yet intended to production deployments and has quite a limited request capacity before it starts the denial of service.

Also, it was recommended to add the optional deployment for **gpt-4-vision-preview** (GPT 4 with Vision), which provides analysis and text descriptions of uploaded images. As of October 2024, this vision model was discontinued. The recommended replacement is GPT-4o, which seamlessly supports all previous features.

- To enable the deployment of Vision-preview in Azure Open AI, you must configure the Azure AI Computer Vision resource in the same Azure region as your GPT-4 with Vision resource.
  - More information: *https://learn.microsoft.com/en-us/azure/ai-services/openai/how-to/gpt-with-vision*
  - By default, this resource uses Price Tier: Free (F0), which supports 20 Calls per minute, 5K Calls per month.
  - You can choose the more capable Price Tier: S1, which supports 10 Calls per second and costs ~1$ per 1000 calls.
- The recommended zone for the deployment is Swedish Central (as of Dec 2023). It supports both types of imaging models Vision and Dalle3 for image generations.

Since September 2024, new models have been supported by the web part (version 1.3+).

  - **gpt-4o**
  - **gpt-4o-mini**

v1.3.4 released in April 2025 added support for newly released models.

  - **gpt-4.1**
  - **gpt-4.1-mini**
  - **gpt-4.1-nano**
  - **o1**
  - **o1-mini**
  - **o3-mini**
  - **o4-mini**

You should add new models to the optional deployments to support corresponding operations.

Chat Web API

If you opt to use a database for storing chat histories, the App Service "ChatWebApi" should be deployed to manage them.

The current version of ChatWebApi (v1.0, as of November 2023) has been developed as an ASP.NET Core (Minimal) Web API project for Visual Studio 2022, based on .NET 7.0.

The standard features available in Visual Studio 2022 Community Edition can be used to deploy the application as an Azure App Service.

- Consider creating a Windows-based service to simplify troubleshooting. You can opt for Linux if you are comfortable with it. For production deployments, it is recommended to use an Application Service Plan S1 or higher.
- Connect to your Azure Portal and go to the App Service you have created.
  - If you have an instance of API Management service, select the option to deploy the Web API to the Azure Application Management service (this is recommended).
  - Save a publishing profile.
- Compile the application and deploy it to the App Service using your saved publishing profile.

Azure API Management

It is recommended to use the Azure API Management Service (APIM) to ensure secure access to Azure OpenAI API. Key benefits of the APIM include:

- The ability to hide the details of user request authentication and authorization.
- Eliminating the necessity to provide the api-key to the client app, as the api-key does not travel with the browser's requests. Instead, the APIM adds the api-key to the request after verifying the client's identity.
- Facilitating seamless updates of endpoints when newer versions of language models become available. Please follow the instructions and examples provided below to configure the APIM endpoints.

Follow the instructions and examples below to configure APIM endpoints.

Named values

Use the blade API Management service > APIs > Named values

Name	Value
AadId	••••••••••••••••••••
AadId-gt	••••••••••••••••••••
AadId-kavoetokkiisa	••••••••••••••••••••
ApiKeyAzureOpenAIService	••••••••••••••••••••
ApiKeyAzureOpenAIService4	••••••••••••••••••••

This is the vault, which stores secure keys inside an API Management service instance.

- You can just add a new value as Secret.
- Optionally, it supports remote access to Azure Key Vault for enhanced security.

You should add the following keys:

- **AadId**: your Azure AD tenant GUID.
  - If you plan to grant access to APIM endpoints for users from multiple Azure AD domains, you can also add the GUIDs of other AAD tenants like AadId-gt, Aad-kavoerokkiisa, etc.

- **ApiKeyAzureOpenAIService**: stores the **api-key** for Azure OpenAI service instance, which contains GPT-3.5 language model deployment.
  - You can find api-key under tenant-specific URLs like https://portal.azure.com/#@**[domain.onmicrosoft.com]**/resource/subscriptions/ **[subscriptionguid]**/resourceGroups/openai/providers/Microsoft.CognitiveServices/acco unts/**[openai-account]**/cskeys
- **ApiKeyAzureOpenAIService4:** the same key as above or a separate **api-key** in another Azure OpenAI instance, which contains GPT-4 deployment (if the first instance only has GPT-3.5).
- **ApiKeyNativeOpenAI:** the optional api-key, which can be used for native open AI deployment(s) in API Management service.

APIs

Use the blade API Management service > APIs > APIs

Create three sets of APIs with settings shown in the table below.

OpenAI

Display name: OpenAI

Name: openai

Description: Azure Open AI Service

Web service URL: https://**your-openai-instance**.openai.azure.com

- Use your instance of [Azure OpenAI service](#).
- Replace **your-openai-instance** with your instance with GPT-3.5 deployment.

URL scheme: HTTPS

API URL suffix: openai

- Base URL: https://**tenant**.azure-api.net/openai
- Replace **tenant** with your instance of API Management service.

Subscription required: **No** (clear default checkbox, it is important!)

Security: User authorization, None (default)

General

* Display name

OpenAI

* Name

openai

Description ❶

Web service URL

https://openai4-ttlopenai-dev.openai.azure.com

URL scheme

○ HTTP   ◉ HTTPS   ○ HTTP(S)

API URL suffix

openai

Base URL

https://apim-ttlopenai-dev.azure-api.net/openai

Tags

Products ❶

No products selected

❶ To publish the API, you must associate it with a product.

Gateways

Managed ×

Subscription

Subscription required

☐

Header name

Ocp-Apim-Subscription-Key

Query parameter name

subscription-key

Security

User authorization   ◉ None   ○ OAuth 2.0   ○ OpenID connect

OpenAI4

Display name: OpenAI4

Name: openai4

Description: Azure Open AI Service

Web service URL: https://**your-openai-instance**.openai.azure.com

- Use your instance of [Azure OpenAI service](#).
- Replace **your-openai-instance** with your instance with GPT-4 deployment.

URL scheme: HTTPS

API URL suffix: openai

- Base URL: https://**tenant**.azure-api.net/openai4
- Replace **tenant** with your instance of API Management service.

Subscription required: **No** (clear default checkbox, it is important!)

Security: User authorization, None (default)

OpenAI-Native (optional)

Display name: OpenAI-Native

Name: openainative

Description: Native Open AI API

Web service URL: https://api.openai.com/v1/chat (without /completions)

URL scheme: HTTPS

API URL suffix: openainative

- Base URL: https://**tenant**.azure-api.net/openainative
- Replace **tenant** with your instance of API Management service.

Subscription required: **No** (clear default checkbox, it is important!)

Security: User authorization, None (default)

ChatWebApi (optional)
Display name: ChatWebApi

Name: chatwebapi

Description: Chat Web Api

Web service URL: https://**your-chatwapi-app-service**.azurewebsites.net

- Use your Chat Web Api App Service deployment.
- Replace **your-chatwapi-app-service**.

URL scheme: HTTPS

API URL suffix: chatwebapi

- Base URL: https://**tenant**.azure-api.net/chatwebapi
- Replace **tenant** with your instance of API Management service.

Subscription required: **No** (clear default checkbox, it is important!)

Security: User authorization, None (default)

After that, publish your App Service for Chat Web Api and deploy it to the configured APIM-endpoint ChatWebApi. You can use Visual Studio 2022 to simplify Publishing and APIM-deployment actions.

Bing (optional)
Note you must have Azure Bing Search service deployed to use this endpoint.

Display name: Bing

Name: bing

Description: Azure Bing Search service

Web service URL: https://api.bing.microsoft.com/v7.0

URL scheme: HTTPS

API URL suffix: bing

- Base URL: https://**tenant**.azure-api.net/bing
- Replace **tenant** with your instance of API Management service.

Subscription required: **No** (clear default checkbox, it is important!)

Security: User authorization, None (default)

Google (optional)

Note you must have Google Custom Search service available in your Google APIs subscription to use this endpoint.

Display name: Google

Name: google

Description: Google Custom Search service

Web service URL: https://www.googleapis.com/customsearch/v1

URL scheme: HTTPS

API URL suffix: google

- Base URL: https://**tenant**.azure-api.net/google
- Replace **tenant** with your instance of API Management service.

Subscription required: **No** (clear default checkbox, it is important!)

Security: User authorization, None (default)

API operations

Subsets of API operations should be created under each of your API.

- Clients will refer to them as https://<baseURL>/<operationurl>
- For example, like https://**tenant**.azure-api.net/openai and https://**tenant**.azure-api.net/openai4

API settings for OpenAI

Configure All operations. Key points are shown below.

- Adjust allowed origins if necessary. Have a look at [Known issues](#).
- AadId and ApiKeyAzureOpenAIService are tokens for values stored in [Named values](#).

A sample working config is given below.

```
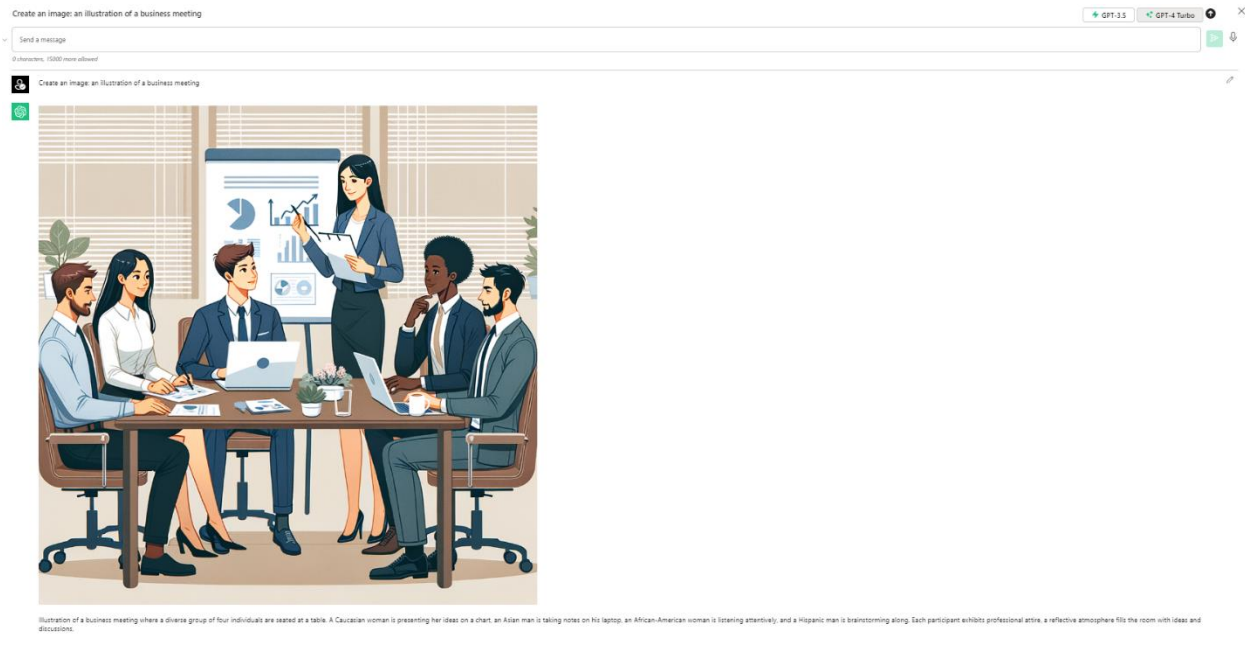<policies>
    <inbound>
        <base />
        <cors allow-credentials="false">
```

```xml
            <allowed-origins>
                <origin>*</origin>
            </allowed-origins>
            <allowed-methods preflight-result-max-age="300">
                <method>GET</method>
                <method>POST</method>
                <method>OPTIONS</method>
            </allowed-methods>
            <allowed-headers>
                <header>Access-Control-Allow-Origin</header>
                <header>Authorization</header>
                <header>Content-Type</header>
                <header>Origin</header>
            </allowed-headers>
        </cors>
        <validate-jwt header-name="Authorization" failed-validation-
httpcode="401" failed-validation-error-message="Unauthorized. Access token is
missing or invalid.">
            <openid-config
url="https://login.microsoftonline.com/organizations/v2.0/.well-known/openid-
configuration" />
            <issuers>
                <issuer>https://sts.windows.net/{{AadId}}/</issuer>
                <!--Optional authorized external domains (end with backslash)-->
                <issuer>https://sts.windows.net/{{AadId-gt}}/</issuer>
                <issuer>https://sts.windows.net/{{AadId-kavoetokkiisa}}/</issuer>
            </issuers>
        </validate-jwt>
        <set-header name="api-key" exists-action="override">
            <value>{{ApiKeyAzureOpenAIService}}</value>
        </set-header>
    </inbound>
    <backend>
        <base />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

After that, you need to add and configure the operations for your OpenAI API.

chat

Display name: chat

Name: chat (it can add any descriptive value, not important)

URL: POST /chat

Description: Chat for GPT-3.5

Display name: chat

Name: chat

URL: POST /chat

Description: Chat for GPT-3.5

Tags: e.g. Booking

Configuration:

- Add a rewrite URL like shown below. It should correspond to backend URL in Azure OpenAI.
    - Clients send POST requests with payload to the APIM's URL https://**tenant**.azure-api.net/openai/chat
    - APIM transforms the URL and forwards the payload to Azure OpenAI service.

- **gpt-35-turbo-16k** is a chat model for GPT-3.5 deployed to your Azure OpenAI service instance.

```
<policies>
    <inbound>
        <base />
        <rewrite-uri template="/openai/deployments/gpt-35-turbo-
16k/chat/completions?api-version=2023-07-01-preview" />
    </inbound>
    <backend>
        <base />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

dalle (optional)

Display name: dalle

Name: dalle (it can add any descriptive value, not important)

URL: POST /dalle

Frontend

* Display name	dalle	
* Name	dalle	
* URL	POST ⌄	/dalle
Description ❶		
Tags	e.g. Booking	

Configuration:

- Add a rewrite URL like shown below. It should correspond to backend URL in Azure OpenAI.
  - Clients send POST requests with payload to the APIM's URL https://**tenant**.azure-api.net/openai/dalle
  - APIM transforms the URL and forwards the payload to Azure OpenAI Dalle-service.

Dalle3 is the newest imaging model, which must be deployed into your Azure OpenAI service instance It is supported, for example, in Swedish Central zone.

```
<policies>
    <inbound>
        <base />
        <rewrite-uri template="/openai/deployments/Dalle3/images/generations?api-version=2023-12-01-preview" />
    </inbound>
    <backend>
        <base />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

Optional new models for the /openai operation: add and configure o1, o1-mini, o3-mini, and o4-mini like shown below

Display name: o1 (o1-mini, o3-mini, o4-mini)

Name: o1 (o1-mini, o3-mini, o4-mini)

URL: POST /o1 (/o1-mini, /o3-mini, /o4-mini)

Description: Chat for o1-mini (o1, o3-mini, o4-mini)

Configuration:

- Add a rewrite URL like shown below. It should correspond to backend URL in Azure OpenAI.
  - Clients send POST requests with payload to the APIM's URL https://**tenant**.azure-api.net/openai/o1
    - https://**tenant**.azure-api.net/openai/o1mini
    - https://**tenant**.azure-api.net/openai/o3mini
    - https://**tenant**.azure-api.net/openai/o4mini

  - APIM transforms the URL and forwards the payload to Azure OpenAI service.

- For example, o1-mini is a chat model deployed to your Azure OpenAI service instance.

```
<policies>
  <inbound>
        <base />
        <rewrite-uri template="/openai/deployments/o1-mini/chat/completions?api-
version=2025-01-01-preview" />
    </inbound>
    <backend>
        <forward-request timeout="180" fail-on-error-status-code="true" buffer-
response="false" />
    </backend>
    <outbound>
```

```
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

One more example for o3-mini.

```
<policies>
  <inbound>
    <base />
    <rewrite-uri template="/openai/deployments/o3-mini/chat/completions?api-version=2025-03-01-preview" />
  </inbound>
  <backend>
    <forward-request timeout="180" fail-on-error-status-code="true" buffer-response="false" />
  </backend>
  <outbound>
    <base />
  </outbound>
  <on-error>
    <base />
  </on-error>
</policies>
```

And one more example for o4-mini, which forwards the request to another Azure OpenAI service instance.

```
<policies>
  <inbound>
    <base />
    <send-request mode="copy" response-variable-name="answer">
      <set-url>https://another-azure-open-ai-instance.openai.azure.com/openai/deployments/o4-mini/chat/completions?api-version=2025-03-01-preview</set-url>
      <set-method>POST</set-method>
      <set-header name="Authorization" exists-action="override">
        <value>Bearer {{ApiKeyAzureOpenAIService-for-another-instance}}</value>
      </set-header>
      <set-header name="Content-Type" exists-action="override">
        <value>application/json</value>
      </set-header>
      <set-header name="Accept" exists-action="override">
        <value>application/json</value>
      </set-header>
    </send-request>
    <return-response response-variable-name="answer" />
  </inbound>
```

```
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
    <backend />
</policies>
```

API settings for OpenAI4

Configure All operations. Key points are shown below.

- Adjust allowed origins if necessary. Have a look at Known issues.
- AadId and ApiKeyAzureOpenAIService**4** are tokens for values stored in Named values.

A sample working config is given below.

```
<policies>
    <inbound>
        <base />
        <cors allow-credentials="false">
            <allowed-origins>
                <origin>*</origin>
            </allowed-origins>
            <allowed-methods preflight-result-max-age="300">
                <method>GET</method>
                <method>POST</method>
                <method>OPTIONS</method>
            </allowed-methods>
            <allowed-headers>
                <header>Access-Control-Allow-Origin</header>
                <header>Authorization</header>
                <header>Content-Type</header>
                <header>Origin</header>
            </allowed-headers>
        </cors>
        <validate-jwt header-name="Authorization" failed-validation-
httpcode="401" failed-validation-error-message="Unauthorized. Access token is
missing or invalid.">
            <openid-config
url="https://login.microsoftonline.com/organizations/v2.0/.well-known/openid-
configuration" />
            <issuers>
                <issuer>https://sts.windows.net/{{AadId}}/</issuer>
                <!--Optional authorized external domains (end with backslash)-->
                <issuer>https://sts.windows.net/{{AadId-gt}}/</issuer>
                <issuer>https://sts.windows.net/{{AadId-kavoetokkiisa}}/</issuer>
            </issuers>
        </validate-jwt>
        <set-header name="api-key" exists-action="override">
```

```
                <value>{{ApiKeyAzureOpenAIService4}}</value>
        </set-header>
    </inbound>
    <backend>
        <base />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

After that, you need to add and configure an operation for your OpenAI**4** API:

chat
Display name: chat

Name: chat (it can add any descriptive value, not important)

URL: POST /chat

Description: Chat for GPT-4

* Display name	chat
* Name	chat
* URL	POST ⌄ /chat
Description ❶	Chat for GPT-4
Tags	e.g. Booking

Configuration:

- Add a rewrite URL like shown below. It should correspond to backend URL in Azure OpenAI.
    - Clients send POST requests with payload to the APIM's URL https://**tenant**.azure-api.net/openai**4**/chat
    - APIM transforms the URL and forwards the payload to Azure OpenAI service.

- **gpt-4-32k** is a chat model for GPT-4 deployed to your Azure OpenAI service instance.

```
<policies>
    <inbound>
        <base />
        <rewrite-uri template="/openai/deployments/gpt-4-
32k/chat/completions?api-version=2023-07-01-preview" />
    </inbound>
```

```
    <backend>
        <base />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

chat4o

Display name: chat4o

Name: chat4o

URL: POST /chat4o

Description: Forwarding to GPT 4o endpoint

Configuration:

- Add a rewrite URL like shown below. It should correspond to backend URL in Azure OpenAI.
  - Clients send POST requests with payload to the APIM's URL https://**tenant**.azure-api.net/openai**4**/chat4o
  - APIM transforms the URL and forwards the payload to Azure OpenAI service.

- **gpt-4o** is a chat model for GPT-4o deployed to your Azure OpenAI service instance.

```
<policies>
 <inbound>
    <base />
    <rewrite-uri template="/openai/deployments/gpt-4o/chat/completions?api-
version=2024-02-15-preview" />
```

```
    </inbound>
    <backend>
        <forward-request timeout="180" fail-on-error-status-code="true" buffer-
response="false" />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

chat4omini

Display name: chat4omini

Name: chat4omini

URL: POST /chat4omini

Description: Forwarding to GPT 4o Mini endpoint

Configuration:

- Add a rewrite URL like shown below. It should correspond to backend URL in Azure OpenAI.
    - Clients send POST requests with payload to the APIM's URL https://**tenant**.azure-api.net/openai**4**/chat4omini
    - APIM transforms the URL and forwards the payload to Azure OpenAI service.

- **gpt-4o-mini** is a chat model for GPT-4o deployed to your Azure OpenAI service instance.

```
<policies>
 <inbound>
     <base />
```

```
        <rewrite-uri template="/openai/deployments/gpt-4o-
mini/chat/completions?api-version=2024-02-15-preview" />
    </inbound>
    <backend>
        <forward-request timeout="180" fail-on-error-status-code="true" buffer-
response="false" />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

chat41 (chat41mini, chat41nano)

Display name: chat41 (chat41mini, chat41nano)

Name: chat41 (chat41mini, chat41nano)

URL: POST /chat41 (/chat41mini, /chat41nano)

Description: Forwarding to GPT 4.1 endpoint

Configuration:

- Add a rewrite URL like shown below. It should correspond to backend URL in Azure OpenAI.
  - Clients send POST requests with payload to the APIM's URL https://**tenant**.azure-api.net/openai**4**/chat41
    - https://**tenant**.azure-api.net/openai**4**/chat41mini
    - https://**tenant**.azure-api.net/openai**4**/chat41nano
  - APIM transforms the URL and forwards the payload to Azure OpenAI service.

- **gpt-4.1** is a chat model for GPT-41 deployed to your Azure OpenAI service instance.
- **gpt-4.1-mini** and **gpt-4.1-nano** should have similar configurations.

```
<policies>
    <inbound>
        <base />
        <rewrite-uri template="/openai/deployments/gpt-4.1/chat/completions?api-
version=2025-03-01-preview" />
    </inbound>
    <backend>
        <forward-request timeout="180" fail-on-error-status-code="true" buffer-
response="false" />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

chatpreview (optional)

This is an additional operation to distinguish the calls to Azure OpenAI deployments of GPT 4 (gpt-4-32k) and GPT 4 Turbo (gpt-4-1106-preview) when both are enabled simultaneously in the web part settings.

Display name: chatpreview

Name: chatpreview (it can add any descriptive value, not important)

URL: POST /chatpreview

Frontend

* Display name	chatpreview	
* Name	chatpreview	
* URL	POST ⌄	/chatpreview
Description ⓘ	This is an additional operation to distinguish the calls to Azure OpenAI deployments of GPT 4 (gpt-4-32k) and GPT 4 Turbo (gpt-4-1106-preview) when both are enabled simultaneously in the web part settings.	
Tags	e.g. Booking	

Configuration:

- Add a rewrite URL like shown below. It should correspond to backend URL in Azure OpenAI.
  - Clients send POST requests with payload to the APIM's URL https://**tenant**.azure-api.net/openai**4**/chatpreview
  - APIM transforms the URL and forwards the payload to Azure OpenAI service.

- **gpt-4-1106-preview** is a chat model for GPT-4 Turbo deployed to your Azure OpenAI service instance.

```
<policies>
    <inbound>
        <base />
        <rewrite-uri template="/openai/deployments/gpt-4-1106-
preview/chat/completions?api-version=2023-07-01-preview" />
    </inbound>
    <backend>
        <forward-request timeout="180" fail-on-error-status-code="true" buffer-
response="false" />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

vision (optional)

This is an additional operation to make calls to the optionally configured Azure OpenAI deployment of GPT 4 Vision (gpt-4-vision-preview).

- *To enable Vision-preview model on Azure OpenAI, you must deploy a* <u>*Computer Vision resource*</u> *as mentioned above (refer to Model deployments).*

Display name: vision

Name: vision (it can add any descriptive value, not important)

URL: POST /vision

Frontend

* Display name	vision	
* Name	vision	
* URL	POST	/vision
Description ⓘ	GPT4-vision-preview	
Tags	e.g. Booking	

Configuration:

- Add a rewrite URL like shown below. It should correspond to backend URL in Azure OpenAI.
  - Clients send POST requests with payload to the APIM's URL https://**tenant**.azure-api.net/openai**4**/vision

- APIM transforms the URL and forwards the payload to Azure OpenAI service.

- **gpt-4-vision-preview** is a chat model for GPT-4 deployed to your Azure OpenAI service instance.

```
<policies>
    <inbound>
        <base />
        <rewrite-uri template="/openai/deployments/gpt-4-vision-
preview/chat/completions?api-version=2023-07-01-preview" />
    </inbound>
    <backend>
        <forward-request timeout="180" fail-on-error-status-code="true" buffer-
response="false" />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

API settings for OpenAI-Native (optional)

Configure All operations. Key points are shown below.

- Adjust allowed origins if necessary. Have a look at <u>Known issues</u>.
- AadId and ApiKeyNativeOpenAI are tokens for values stored in <u>Named values</u>.

A sample working config is given below.

```
<policies>
    <inbound>
        <base />
        <cors allow-credentials="false">
            <allowed-origins>
                <origin>*</origin>
            </allowed-origins>
            <allowed-methods preflight-result-max-age="300">
                <method>GET</method>
                <method>POST</method>
                <method>OPTIONS</method>
            </allowed-methods>
            <allowed-headers>
                <header>Access-Control-Allow-Origin</header>
                <header>Authorization</header>
                <header>Content-Type</header>
                <header>Origin</header>
            </allowed-headers>
        </cors>
```

```xml
        <validate-jwt header-name="Authorization" failed-validation-
httpcode="401" failed-validation-error-message="Unauthorized. Access token is
missing or invalid.">
            <openid-config
url="https://login.microsoftonline.com/organizations/v2.0/.well-known/openid-
configuration" />
            <issuers>
                <issuer>https://sts.windows.net/{{AadId}}/</issuer>
                <!--Optional authorized external domains (end with backslash)-->
                <issuer>https://sts.windows.net/{{AadId-gt}}/</issuer>
                <issuer>https://sts.windows.net/{{AadId-kavoetokkiisa}}/</issuer>
            </issuers>
        </validate-jwt>
        <set-header name="Authorization" exists-action="override">
            <value>Bearer {{ApiKeyNativeOpenAI}}</value>
        </set-header>
    </inbound>
    <backend>
        <base />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

After that, you need to add and configure an operation for your OpenAI-Native API:

chat
Display name: chat

Name: chat (it can add any descriptive value, not important)

URL: POST /chat

Description: Chat for native Open AI

* Display name	chat
* Name	chat
* URL	POST ∨ /chat
Description ⓘ	
Tags	e.g. Booking

Configuration:

- Add a rewrite URL like shown below.
  - Clients send POST requests with payload to the APIM's URL https://**tenant**.azure-api.net/openainative/chat
  - APIM transforms the URL and forwards the payload to native OpenAI API.
  - Transformed URL corresponds to https://api.openai.com/v1/chat/completions

```
<policies>
    <inbound>
        <base />
        <rewrite-uri template="/completions" />
    </inbound>
    <backend>
        <forward-request timeout="120" fail-on-error-status-code="true"
buffer-response="false" />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

dalle (optional)

This is an optional configuration to support Dall-e-3 image generation model of Native OpenAI.

Display name: dalle

Name: dalle (it can add any descriptive value, not important)

URL: POST /dalle

Frontend

* Display name	dalle	
* Name	dalle	
* URL	POST ⌄	/dalle
Description ⓘ		
Tags	e.g. Booking	

Configuration:

- Add a rewrite URL like shown below. It should correspond to backend URL in Native OpenAI.

- Clients send POST requests with payload to the APIM's URL https://**tenant**.azure-api.net/openainative/dalle
- APIM transforms the URL and forwards the payload to Dalle-service of the Native OpenAI.
- The transformed URL corresponds to https://api.openai.com/v1/images/generations

```
<policies>
    <inbound>
        <base />
        <rewrite-uri template="/images/generations" />
    </inbound>
    <backend>
        <base />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

tts (optional)

This is an optional configuration to support text-to-speech model of Native OpenAI (tts-1).

Display name: tts

Name: tts (it can add any descriptive value, not important)

URL: POST /tts

Frontend

* Display name	tts	
* Name	tts	
* URL	POST	/tts
Description 🛈		
Tags	e.g. Booking	

Configuration:

- Add a rewrite URL like shown below. It should correspond to backend URL in Native OpenAI.
  - Clients send POST requests with payload to the APIM's URL https://**tenant**.azure-api.net/openainative/tts

- APIM transforms the URL and forwards the payload to text-to-speech service of the Native OpenAI.
- The transformed URL corresponds to https://api.openai.com/v1/audio/speech

```
<policies>
    <inbound>
        <base />
        <rewrite-uri template="/audio/speech" />
    </inbound>
    <backend>
        <base />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

API settings for ChatWebApi (optional)

Configure All operations. Key points are shown below.

- Adjust allowed origins if necessary. Have a look at Known issues.
- AadId- are tokens for values stored in Named values.

A sample working config is given below.

```
<policies>
    <inbound>
        <base />
        <cors allow-credentials="false">
            <allowed-origins>
                <origin>*</origin>
            </allowed-origins>
            <allowed-methods preflight-result-max-age="300">
                <method>GET</method>
                <method>POST</method>
                <method>PUT</method>
                <method>DELETE</method>
                <method>OPTIONS</method>
            </allowed-methods>
            <allowed-headers>
                <header>Access-Control-Allow-Origin</header>
                <header>Authorization</header>
                <header>Content-Type</header>
                <header>Origin</header>
            </allowed-headers>
        </cors>
```

```
        <validate-jwt header-name="Authorization" failed-validation-
httpcode="401" failed-validation-error-message="Unauthorized. Access token is
missing or invalid.">
            <openid-config
url="https://login.microsoftonline.com/organizations/v2.0/.well-known/openid-
configuration" />
            <issuers>
                <issuer>https://sts.windows.net/{{AadId}}/</issuer>
                <!--Optional authorized external domains (end with backslash)-->
                <issuer>https://sts.windows.net/{{AadId-gt}}/</issuer>
                <issuer>https://sts.windows.net/{{AadId-kavoetokkiisa}}/</issuer>
            </issuers>
        </validate-jwt>
    </inbound>
    <backend>
        <base />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

API settings for Bing (optional)

Configure All operations. Key points are shown below.

- Adjust allowed origins if necessary. Have a look at Known issues.
- AadId and ApiKeyBingSearch are tokens for values stored in Named values.
  - ApiKeyBingSearch stores a Key for your Azure Bing Search Service

A sample working config is given below.

```
<policies>
    <inbound>
        <base />
        <cors allow-credentials="false">
            <allowed-origins>
                <origin>*</origin>
            </allowed-origins>
            <allowed-methods preflight-result-max-age="300">
                <method>GET</method>
            </allowed-methods>
            <allowed-headers>
                <header>Access-Control-Allow-Origin</header>
                <header>Authorization</header>
                <header>Content-Type</header>
                <header>Origin</header>
            </allowed-headers>
```

```
        </cors>
        <validate-jwt header-name="Authorization" failed-validation-
httpcode="401" failed-validation-error-message="Unauthorized. Access token is
missing or invalid.">
            <openid-config
url="https://login.microsoftonline.com/organizations/v2.0/.well-known/openid-
configuration" />
            <issuers>
                <issuer>https://sts.windows.net/{{AadId}}/</issuer>
                <!--Optional authorized external domains (end with backslash)-->
                <issuer>https://sts.windows.net/{{AadId-gt}}/</issuer>
                <issuer>https://sts.windows.net/{{AadId-kavoetokkiisa}}/</issuer>
            </issuers>
        </validate-jwt>
        <set-header name="Ocp-Apim-Subscription-Key" exists-action="override">
            <value>{{ApiKeyBingSearch}}</value>
        </set-header>
    </inbound>
    <backend>
        <base />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

After that, you need to add and configure the search operation for your Bing endpoint:

search

Display name: search

Name: search (it can add any descriptive value, not important)

URL: GET /search

Frontend

* Display name	search
* Name	search
* URL	GET	/search
Description ❶	
Tags	e.g. Booking

Configuration:

- Add a rewrite URL like shown below.
  - Clients send GET requests with query string parameters to the APIM's URL
    https://**tenant**.azure-api.net/bing/search?q=QUERY_TEXT&mkt=LOCALE
  - APIM transforms the URL and forwards the payload to Azure Bing Search Service.
  - Transformed URL corresponds to
    https://api.bing.microsoft.com/v7.0/search?q=QUERY_TEXT&mkt=LOCALE

- Use default APIM policies.

API settings for Google (optional)

Configure All operations. Key points are shown below.

- Adjust allowed origins if necessary. Have a look at Known issues.
- AadId- are tokens for values stored in Named values.

ApiKeyGoogleSearch stores a query string combination of **key=API_KEY&cx=SEARCH_ENGINE_ID** used in your instance of Google Custom Search API. Please refer to Google documentation for Custom Search to generate those values (https://developers.google.com/custom-search/v1/introduction). They are available free of charge to developers.

A sample working config is given below.

- Note that unlike the Bing endpoint the main policy of Google endpoint does not use the header with ApiKeyGoogleSearch. In Google search, the key is injected by APIM directly into a query string in the search operation (as shown below).

```
<policies>
    <inbound>
        <base />
        <cors allow-credentials="false">
            <allowed-origins>
                <origin>*</origin>
            </allowed-origins>
            <allowed-methods preflight-result-max-age="300">
                <method>GET</method>
            </allowed-methods>
            <allowed-headers>
                <header>Access-Control-Allow-Origin</header>
                <header>Authorization</header>
                <header>Content-Type</header>
                <header>Origin</header>
            </allowed-headers>
        </cors>
        <validate-jwt header-name="Authorization" failed-validation-
httpcode="401" failed-validation-error-message="Unauthorized. Access token is
missing or invalid.">
```

```
            <openid-config
url="https://login.microsoftonline.com/organizations/v2.0/.well-known/openid-
configuration" />
            <issuers>
                <issuer>https://sts.windows.net/{{AadId}}/</issuer>
                <!--Optional authorized external domains (end with backslash)-->
                <issuer>https://sts.windows.net/{{AadId-gt}}/</issuer>
                <issuer>https://sts.windows.net/{{AadId-kavoetokkiisa}}/</issuer>
            </issuers>
        </validate-jwt>
    </inbound>
    <backend>
        <base />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

After that, you need to add and configure the search operation for your Google endpoint:

search

Display name: search

Name: search (it can add any descriptive value, not important)

URL: GET /search

Frontend

* Display name	search	
* Name	search	
* URL	GET	/search
Description ⓘ		
Tags	e.g. Booking	

Configuration:

- Add a rewrite URL like shown below.
    - Clients send GET requests with query string parameters to the APIM's URL
      https://**tenant**.azure-api.net/google/search?q=QUERY_TEXT&lr=LOCALE
    - APIM transforms the URL and forwards the payload to Google Custom Search service.

- o Transformed URL corresponds to https://www.googleapis.com/customsearch/v1?**key=API_KEY&cx=SEARCH_ENGINE_ID**&q=QUERY_TEXT&lr=LOCALE
  - The first two parameters should be injected within the APIM policy as shown below.

```
<policies>
    <inbound>
        <base />
        <rewrite-uri template="?{{ApiKeyGoogleSearch}}" />
    </inbound>
    <backend>
        <base />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

Known issues

There was an intermittent issue that might have been associated with the use of the Consumption plan for API Management in the DEV-tenant.

- Please refer to the table available at https://azure.microsoft.com/en-us/pricing/details/api-management/?ef_id=_k_CjwKCAjw67ajBhAVEiwA2g_jEHK13rhWiOba6Xz7YiLxkWCKMivEjoXqVmi8dQPCK3rVkmewyqSb6BoCrVkQAvD_BwE_k_&OCID=AIDcmmftanc7uz_SEM__k_CjwKCAjw67ajBhAVEiwA2g_jEHK13rhWiOba6Xz7YiLxkWCKMivEjoXqVmi8dQPCK3rVkmewyqSb6BoCrVkQAvD_BwE_k_&gclid=CjwKCAjw67ajBhAVEiwA2g_jEHK13rhWiOba6Xz7YiLxkWCKMivEjoXqVmi8dQPCK3rVkmewyqSb6BoCrVkQAvD_BwE#pricing
- As indicated there, the Consumption plan uses Shared isolation.
- According to discussions in Microsoft forums, this might occasionally cause CORS-conflicts if you specify explicit values in allowed-origins.
  - o Other plans offer Private isolation and do not seem to encounter similar issues.
- Anyway, it might be prudent to avoid defining explicit CORS rules in APIM policies.

Security

Frontend

The web part is hosted on SharePoint Online and utilizes the standard authentication options provided by the platform. Connection to backend endpoints is established using the Azure App registration **openaiwp** as outlined in the "Permissions" chapter of this document.

Backend

For more information on securing backend services, please refer to the supplementary document **azure-openai-chat-security.pdf.**