IntroductionJanuary 26, 2026

Who are we?

CreateAI is a platform designed to support advanced interactions with large language models (LLMs). The platform supports 25+ large language models capable of audio, image and text modality. The documentation provided here aims to offer all necessary details about how to effectively use our APIs, including descriptions of payloads, parameters, response formats, and full API specifications.

Note

The DocuBot link on the far right is available for you to chat to our bot about the documentation or any other API related help. If you get a 404 error, kindly contact Siddharth Jain at sjain238@asu.edu regarding access to the bot. Incase the bot does not work well for you feel free to contact your administrator regarding further assistance. Thank you!

Warning

This documentation is still under development and might be incomplete. Please contact your administrator incase something goes wrong.

Usage

Token

To use our platform, you will first need a token. Please contact Ayat Sweid (Ayat.Sweid@asu.edu) or Paul Alvarado (Paul.Alvarado.1@asu.edu) for API access token. You will also be given an API for your department or team to access the platform.

Request Method

Our platform supports various methods depending on what you want to do. We support both **REST** and **WebSocket** methods for most of the methods, which will

be detailed in the corresponding method section below. After the August 29th update, we have a new way of calling various methods. The method in the URL stays `queryV2` for REST and action stays `queryV2` for WebSocket, but with an additional parameter in the payload called `endpoint`, which can be one of the following: `"speech"`, `"image"`, `"vision"`, or `"audio"`.

URLs

- **REST:**POC `https://api-main-poc.aiml.asu.edu/{method}` Beta `https://api-main-beta.aiml.asu.edu/{method}` Prod `https://api-main.aiml.asu.edu/{method}` Your access token goes into the authorization header as a bearer token.
- **WebSocket:**POC `wss://apiws-main-poc.aiml.asu.edu/?access_token={access_token}` Beta `wss://apiws-main-beta.aiml.asu.edu/?access_token={access_token}` Prod `wss://apiws-main.aiml.asu.edu/?access_token={access_token}`

Note: The default value for most users is `main`, but teams such as LE, KE, or Edplus may have their own custom API. The `method` parameter can be one of the following: `query` or `queryV2`, `search`, `embeddings`, or `chunk`.

Project Owner Token

The project owner token allows you to modify your project settings, upload files, manage the knowledge base, and more.

File Upload API

To upload a file to your knowledge base, use the following API call:

```
            PROD - POST https://api-main.aiml.asu.edu/project
            BETA - POST https://api-main-beta.aiml.asu.edu/pr
oject
            POC  - POST https://api-main-poc.aiml.asu.edu/pro
ject

            Content-Type: application/json
            Authorization: Bearer <Your_Project_Owner_Token>

            {
```

```
  "resource": "data",
  "method": "upload",
  "details": {
    "project_id": "your_project_id_here",
    "db_type": "opensearch",
    "files": [
      {
        "file_name": "your_file_name.ext",
        "search_tags": ["list_of_tags"],
        "metadata": {
          "any_metadata_field": "value"
        },
        "selected": true,
        "visible": true
      }
    ]
  }
}
```

This call will return a pre-signed URL that you can use to upload your file.

Example in Python

Below is a complete Python script demonstrating file upload, status checking, and querying via WebSocket.

Check Upload Status (Optional)

To check the status of uploaded files, use the same URL with the following payload:

```
POST https://api-main-{env}.aiml.asu.edu/project
where {env} - poc/beta/prod

Content-Type: application/json
Authorization: Bearer <Your_Project_Owner_Token>
```

```
{
  "resource": "data",
  "method": "list",
  "details": {
    "project_id": "your_project_id_here",
    "db_type": "opensearch"
  }
}
```

This request is optional and only returns the status of uploaded files.

Authentication

This API uses Bearer Token for authentication. The token should be included in the Authorization header of the request.

```
Authorization: Bearer <Your_Token_Here>
```

Common Issues and Troubleshooting

- **Invalid key-value pair in Authorization header:** Ensure that the token is correctly included and formatted. It should be in the form of a Bearer token.
- **400 Bad Request:** Check if the request body or headers are formatted correctly. Specifically, ensure that all required fields are present and correct.
- **Authorization Issues:** If the token does not have access to the requested resources, you might receive errors related to missing permissions. Ensure the correct token is being used.
- **Incorrect Method (POST/GET):** Some endpoints expect a POST request, not a GET. Ensure the correct HTTP method is used for each endpoint.

Token Information

Most of the tokens we provide is linked to your project ID, so all payload parameters will be used via the project settings unless you override them. Any search will be done against the project ID.

For example, if your project ID is `your_project_id_here`, the token will be linked to that project and you can check on https://jwt.io.

Methods

1. POST /search

Perform a search operation.

Example Query:

```
        {
          "query": "What is your name?",
          "search_params": {
            "collection": "your_project_id_here",
            # or collection name if you have a custom collect
ion
            "source_name": "list of your file names if you wa
nt to filter",
            "top_k": "number of search results to retrieve",
            "reranker": "boolean",
            "output_fields": ["source_name", "tags", "conten
t", "score"],
            # or whatever you want from query payload
            "retrieval_type": "chunk" or "neighbor" or "docum
ent",
            "expr": "source_name != 'File_name_1' && source_n
ame != 'File_name_2'"
          }
        }
```

Parameters:

- `source_name` (array) (Optional):

  e.g. ["file_name1.pdf", "file_name2.txt"]

  You can use this to filter your data before searching with the file names you provided while uploading the data.

- `top_k` (int, default is 3):

  Number of results to retrieve. You can set the number of chunks or documents you want to retrieve here. It can range from 1 to any desired number. Keep in mind, setting a higher number may slow the response time and sometimes reduce quality.

- `output_fields` (array) (Optional):

  e.g. ["source_name", "content", "source_type"]

  This lets you get the metadata at the end of a response. You need to set the response format to JSON in the additional parameters.

- `retrieval_type` (string, default: "chunk"):

  Options are: 'chunk', 'neighbor', 'document'. This defines how the data is retrieved.

- `expr` (string) (Optional):

  Custom expression to filter results, e.g. "metadata1 == 'corresponding_filter' && metadata2 == 'corresponding_filter'".

2. POST /query

Execute a query with specific parameters.

Example Query:

```
        {
          "action": "query",
          "request_source": "override_params",
          # if you want to override any params set in your pr
oject with the ones you send via the payload below.
          "query": "What is this?",
          "model_provider": "model_provider",
          "model_name": "model_name",
          "session_id": "testingsid123456",
          "model_params": {
            "temperature": "model_params['temperature']",
            "system_prompt": "system_prompt",
```

```
            "top_p": "model_params['top_p']",
            "top_k": "model_params['top_k']",
            "tools": [
              {
                "type": "function",
                "function": {
                  "name": "get_weather",
                  "description": "Get current temperature for
a given location.",
                  "parameters": {
                    "type": "object",
                    "properties": {
                      "location": {
                        "type": "string",
                        "description": "City and country e.g.
Bogotá, Colombia"
                      }
                    },
                    "required": ["location"],
                    "additionalProperties": false
                  },
                  "strict": true
                }
              },
              {
                "type": "function",
                "function": {
                  "name": "get_continent",
                  "description": "Get current continent for a
given location.",
                  "parameters": {
                    "type": "object",
                    "properties": {
                      "location": {
                        "type": "string",
                        "description": "City and country e.g.
```

```
Bogotá, Colombia"
                    }
                  },
                  "required": ["location"],
                  "additionalProperties": false
                },
                "strict": true
              }
            }
          ],
          "enable_search": true,
          "search_params": {
            "collection": "collection",
            "source_name": "source_name list",
            "top_k": "top_k",
            "reranker": "reranker",
            "retrieval_type": "retrieval_type",
            "output_fields": ["content", "source_name", "pa
ge_number", "source_type"],
            "prompt_mode": "unrestricted",
            "rerank": true,
            "reranker_model": "amazon_rerank",
            "reranker_provider": "aws",
            "top_n": 3
          },
          "history": [
            {
              "query": "What is your name?",
              "response": "My name is DocuBot."
            },
            {
              "query": "Is AI the solution for everythin
g?",
              "response": "No, AI is not the solution for e
verything."
            }
```

```
            ],
            "enable_history": true,
            "response_format": {"type": "json"},
            "enhance_prompt": {"timezone": "MST", "time": tru
e, "date": true, "verbosity": "brief"}
          }
        }
```

Parameters:

- `expr` (string) (Optional):

  Defines custom expression filters (e.g., "tags != 'Module 0' && tags != 'Syllabus'").

- `source_name` (array) (Optional):

  A list of file names (e.g., ["file_name1.pdf", "file_name2.txt"]). Use this to filter your data before searching with the file names you provided while uploading the data.

- `top_k` (int, default is 3):

  Set the number of chunks or documents you want to retrieve. This can range from 1 to the desired number. Keep in mind, higher numbers may slow response time and reduce quality.

- `output_fields` (array) (Optional):

  e.g., ["source_name", "content", "source_type"]. This lets you get the metadata at the end of a response. Set the response format to JSON in the additional parameters.

- `retrieval_type` (string, default: "chunk"):

  Options are: 'chunk', 'neighbor', 'document'. This defines how the data is retrieved.

- `prompt_mode` (string):

  e.g., "restricted", "unrestricted", or "custom". Defines the mode for the prompt used in search queries.

- `search_prompt` (string) (Required when `prompt_mode` is set to "custom"):

Custom search prompt when prompt_mode is set to custom (e.g., "This is your new prompt when search is enabled.")

- `enhance_prompt` (object):

  Parameters to adjust timezone, time, date, and verbosity in the model's responses.

  - `timezone` (string, default: "MST"): Sets the timezone (e.g., "MST").
  - `time` (boolean, default: false): If true, the model will consider the current time in its responses.
  - `date` (boolean, default: false): If true, the model will include the current date in its responses.
  - `verbosity` (string, default: none): Can be set to "brief", "detailed", or "succinct" to control the level of detail in the responses.

- `enable_search` (boolean, default: false):

  If true, enables the search functionality and allows the model to search through your documents.

- `enable_history` (boolean, default: false):

  If true and a session ID is provided, enables the model to use the history from the current session.

Models List

ID	Provider	Is Active	Capabilities
bison	gcp	No	query
bison-32k	gcp	No	query
claude2	aws	No	query
claude2_1	aws	No	query
claude3_haiku	aws	Yes	query, vision
claude3_opus	aws	No	query, vision
claude4_opus	aws	Yes	query, vision
claude4_1_opus	aws	Yes	query, vision

ID	Provider	Is Active	Capabilities
claude4_5_sonnet	aws	Yes	query, vision
claude4_5_haiku	aws	Yes	query, vision
claude4_5_opus	aws	Yes	query, vision
claude3_sonnet	aws	Yes	query, vision
claude3_5_sonnet	aws	Yes	query, vision
claude3_7_sonnet	aws	Yes	query, vision
claude4_sonnet	aws	Yes	query, vision
claudeinstant	aws	No	query
gpt-oss-20b	aws	Yes	query
gpt-oss-120b	aws	Yes	query
command	aws	No	query
commandlight	aws	No	query
geminipro	gcp-deepmind	No	query
geminipro1_5	gcp-deepmind	No	query, vision, audio
geminipro2_5	gcp-deepmind	Yes	query, vision, audio
geminiflash1_5	gcp-deepmind	No	query, vision, audio
geminiflash2	gcp-deepmind	Yes	query, vision, audio
geminiflash2_5	gcp-deepmind	Yes	query, vision, audio
geminiflash3	gcp-deepmind	Yes	query, vision, audio
geminipro3	gcp-deepmind	Yes	query, vision, audio
geminiflash2-lite	gcp-deepmind	Yes	query, vision, audio
geminiflash2_5-lite	gcp-deepmind	Yes	query, vision, audio
geminiflash2-pro	gcp-deepmind	No	query, vision, audio
geminiflash1_5-8b	gcp-deepmind	No	query, vision, audio
imagen3	gcp-deepmind	Yes	image
gpt3_5	openai	Yes	query
gpt3_5-16k	openai	Yes	query
gpt4	openai	Yes	query

ID	Provider	Is Active	Capabilities
gpt4-32k	openai	No	query
gpt4turbo	openai	Yes	query
gpt4o	openai	Yes	query, vision
gpt5	openai	Yes	query, vision
gpt5_chat	openai	Yes	query, vision
gpt5_mini	openai	Yes	query, vision
gpt5_nano	openai	Yes	query, vision
gpt5_1	openai	Yes	query, vision
gpt5_1_instant	openai	Yes	query, vision
gpt5_1_thinking	openai	Yes	query, vision
gpt5_2	openai	Yes	query, vision
gpt5_2_pro	openai	Yes	query, vision
gpt4_1	openai	Yes	query, vision
gpt4_1-mini	openai	Yes	query, vision
gpt4_1-nano	openai	Yes	query, vision
gpt4o_realtime	openai	Yes	realtime
gpt4o_mini_realtime	openai	Yes	realtime
gpt4o_mini	openai	Yes	query, vision
o1-mini	openai	No	query
o3-mini	openai	Yes	query
o3-mini-high	openai	Yes	query
o4-mini-high	openai	Yes	query, vision
o4-mini	openai	Yes	query, vision
o3	openai	Yes	query, vision
o1	openai	Yes	query, vision
tts1	openai	Yes	speech
tts1hd	openai	Yes	speech
gpt4o_mini-tts	openai	Yes	speech

ID	Provider	Is Active	Capabilities
dalle3	openai	Yes	image
whisper-1	openai	Yes	audio
gpt4o-transcribe	openai	Yes	audio
gpt4o_mini-transcribe	openai	Yes	audio
j2mid	aws	No	query
j2ultra	aws	No	query
llama3-8b	aws	Yes	query
llama3-70b	aws	Yes	query
llama3-405b	aws	Yes	query
llama3_3-70b	aws	No	query
llama3_2-1b	aws	Yes	query
llama3_2-3b	aws	Yes	query
llama3_2-11b	aws	Yes	query, vision
llama3_2-90b	aws	Yes	query, vision
llama4_maverick-17b	aws	Yes	query, vision
llama4_scout-17b	aws	Yes	query, vision
mistral-7b	aws	Yes	query
mistral-8×7b	aws	Yes	query
mistral-large	aws	Yes	query
titang1express	aws	Yes	query
titang1lite	aws	No	query
lambda	lambda	Yes	
phi_3_mini_128k	lambda	Yes	query
allam_7b	lambda	Yes	query
llama_3_8b_finetuned	lambda	Yes	query
gatesgrant-4-llama-3B-dslbio	lambda	Yes	query
gatesgrant-1-l3b-context-fin	lambda	Yes	query
gates-2-l3b-fin_pii	lambda	Yes	query

ID	Provider	Is Active	Capabilities
gates-3-l3b-dslbio_pii	lambda	Yes	query
gatesgrant-l3b-base-model	lambda	Yes	query
gates-l3b-fin-pii-con	lambda	Yes	query
gates-l3b-bio-pii-con	lambda	Yes	query
llama_3_8b_base	lambda	Yes	query
llama_guard_8b	lambda	Yes	query
whisper_lambda	lambda	Yes	audio
jina	lambda	Yes	embeddings
nova-lite	aws	Yes	query, vision
nova-pro	aws	Yes	query, vision
nova-micro	aws	Yes	query
nova-premier	aws	Yes	query, vision
cohere_rerank-3_5	aws	Yes	query, rerank
amazon_rerank	aws	Yes	query, rerank
Prompt-Guard-86M	lambda	Yes	query
Prompt-Guard-2-86M	lambda	Yes	query
fyi_speech	fyi	Yes	speech
fyi_chat	fyi	Yes	query
fyi_realtime	fyi	Yes	realtime

Audio Model Example

```
{
  "action": "query",
  "endpoint": "audio",
  "query": "What is this audio?",
  "audio_file": "Base64_encoded_audio_data",
  "model_provider": "openai",
  "model_name": "whisper-1",
  "model_params": {
```

```
        "translation": true,
        "language": "en"
      },
      "response_format": {
        "type": "json"
      }
    }
```

Note: The `audio` model takes input a audio file understands it and translate it to the desired and supported language or responds based on the query asked.

model_params (dict) (Optional): This contains the model-specific parameters. You can set the `translate` boolean here (default is False). This is only applicable for the Whisper model (see payload for example). Additionally, a `language` parameter has been added to specify the language for transcriptions (default is "en" for English ISO Code). See **Language** for various ISO codes and supported languages.

Language (ISO Code) Examples:

- Afrikaans (af), Arabic (ar), Armenian (hy), Azerbaijani (az), Belarusian (be), Bosnian (bs), Bulgarian (bg), Catalan (ca), Chinese (zh), Croatian (hr), Czech (cs), Danish (da), Dutch (nl), English (en), Estonian (et), Finnish (fi), French (fr), Galician (gl), German (de), Greek (el), Hebrew (he), Hindi (hi), Hungarian (hu), Icelandic (is), Indonesian (id), Italian (it), Japanese (ja), Kannada (kn), Kazakh (kk), Korean (ko), Latvian (lv), Lithuanian (lt), Macedonian (mk), Malay (ms), Marathi (mr), Maori (mi), Nepali (ne), Norwegian (no), Persian (fa), Polish (pl), Portuguese (pt), Romanian (ro), Russian (ru), Serbian (sr), Slovak (sk), Slovenian (sl), Spanish (es), Swahili (sw), Swedish (sv), Tagalog (tl), Tamil (ta), Thai (th), Turkish (tr), Ukrainian (uk), Urdu (ur), Vietnamese (vi), Welsh (cy).

Vision Model Example

```
    {
      "action": "query",
      "endpoint": "vision",
      "query": "What is in this image?",
      "image_file": "Base64_encoded_image_data",
      "model_provider": "openai",
```

```
  "model_name": "gpt4o",
  "response_format": {
    "type": "json"
  }
}
```

Note: The `vision` model takes input a image file understands it and describes the image or responds based on what's asked in the query.

Speech Model Example

```
{
  "action": "query",
  "endpoint": "speech",
  "query": "Say Hello in French.",
  "voice": "alloy",
  "model_provider": "openai",
  "model_name": "tts1"
}
```

Note: The `voice` parameter (default is "alloy") accepts: "alloy", "nova", "echo", "fable", "onyx", "shimmer".

Image Generation Model Example

```
{
  "action": "query",
  "endpoint": "image",
  "query": "Generate image of an ice cream",
  "model_provider": "openai",
  "model_name": "dalle3"
}
```

Reranker Model Example

```
      {
        "query": "query",
        "model_provider": "aws",
        "model_name": "reranking_models",
        "model_params": {
          "top_n": 3,
          "documents": [ "list of string of your search resul
ts here or anything you want to rerank" ]
        }
      }
```

/embeddings method

```
      {
          "query": "Hi what is your name?",
          "embeddings_provider": "openai",
          "embeddings_model": "ada",
          "dimensions": 1356,
          "normalize": true
      }
```

Parameters:

- `query` (string): Your query for getting the embeddings.
- `embeddings_provider` (string): Options include "openai", "aws", "azure", "gcp". Specifies the provider for the embeddings.
- `embeddings_model` (string): The corresponding model for the selected provider.
- `dimensions` (int, default is 1024):

  For example, 256, 512, or 1024 (only applicable for the titan2 model).
- `normalize` (boolean, default is false):

  Whether to normalize the embeddings (only applicable for the titan2 model).

Supported Models

The following models are supported across different providers:

- **model_provider = openai, model_name = te3l**
- **model_provider = openai, model_name = ada**
- **model_provider = openai, model_name = te3s**
- **model_provider = aws, model_name = titan**
- **model_provider = aws, model_name = titan2**
- **model_provider = aws, model_name = ce-english**
- **model_provider = aws, model_name = ce-multilingual**
- **model_provider = gcp, model_name = gecko**
- **model_provider = gcp, model_name = gecko-2**
- **model_provider = gcp, model_name = gecko-3**
- **model_provider = gcp, model_name = gecko-multilingual**

/chunks method

This endpoint shows how your document will be chunked and tokenized. It supports various chunking strategies and displays the resulting tokens.

```
            {
              "text": "Text you want to chunk and tokenize",
              "enable_chunks": true,
              "enable_tokenizer": true,
              "chunk_params": {
                "chunk_strategy": "character",  // Options:
"character", "words", "sentence"
                "chunk_size": 256
              },
              "tokenizer_params": {
                "tokenizer_provider": "openai",
                "tokenizer_name": "gpt2"  // or another suppo
rted model
```

```
            }
        }
```

Parameters:

- `text` (string):

  The text that you want to chunk and tokenize.

- `enable_chunks` (boolean):

  Whether to enable chunking of the document (True/False).

- `enable_tokenizer` (boolean):

  Whether to enable tokenization of the document (True/False).

- `chunk_params` (object):

  Defines how the text will be chunked.

  - `chunk_strategy` (string): Options are "character", "words", or "sentence". Defines the strategy used for chunking.
  - `chunk_size` (int): The size of each chunk (e.g., 256 characters, words, or sentence tokens).

- `tokenizer_params` (object):

  Defines the tokenizer configuration.

  - `tokenizer_provider` (string): The provider for tokenization (e.g., "openai").
  - `tokenizer_name` (string): The specific tokenizer model to use (e.g., "gpt2").

Additional Information

When testing an endpoint (for example using Postman):

1. Set the method to POST.
2. Use the URL:
   - For the query endpoint: `https://api-main-beta.aiml.asu.edu/query`
   - For the search endpoint: `https://api-main-beta.aiml.asu.edu/search`
3. In the Headers section, add: `Authorization: Bearer <Your_Token_Here>`

4. In the Body section, select raw and JSON, then add your payload.

Important: Always verify that you are using the correct HTTP method, a valid Bearer token, and that all required parameters are provided.

Additional Features

The query endpoint supports audio, vision, and speech models. By adding the parameter `"request_source": "override_params"` in your payload, you can override any project settings. This allows you to provide custom system prompts and adjust model parameters that may otherwise be set on the platform.

Adding the parameter "request_source": "override_params" you can override any project settings you have enabled. This means that even if you have a system prompt in your project settings, you can override it by sending this flag and the new system prompt in the payload. You can also override the models and use any models that we offer and not available to select via the UI for example, speech, audio and vision models.