OpenAI

Connected System Plugin for **Appian**

Julian Grunauer
Technology Strategy Engineer

Version 1.2.2

appian

Introduction

The introduction of ChatGPT has shaken the world with its revolutionary approach to AI and natural language processing. It is evident that as technology companies move forward integrating AI into their workflow is critical. Appian itself is integrating AI in a variety of ways; from utilizing it as an inspiration tool to being a developer aid.

The OpenAI Connected System allows users to give prompts and receive AI generated responses, whether that be images (DALLE-2), audio (Whisper), or text (ChatGPT). These responses can then be edited, updated, expanded upon, or deleted by the AI model as desired. Users can also fine-tune (custom train) a model based on Record data.

Privacy Policy

All information passed through AI tools will be processed and may remain with the organizations that develop those tools. Please exercise caution with what information is disclosed to the AI tool for this reason.

Further Information

Please see the below resources from OpenAI for any legal questions and concerns.

1. Privacy Policy
2. Terms of Use
3. Sharing & Publication Policy
4. Coordinated Vulnerability Disclosure Policy

Integration Overview

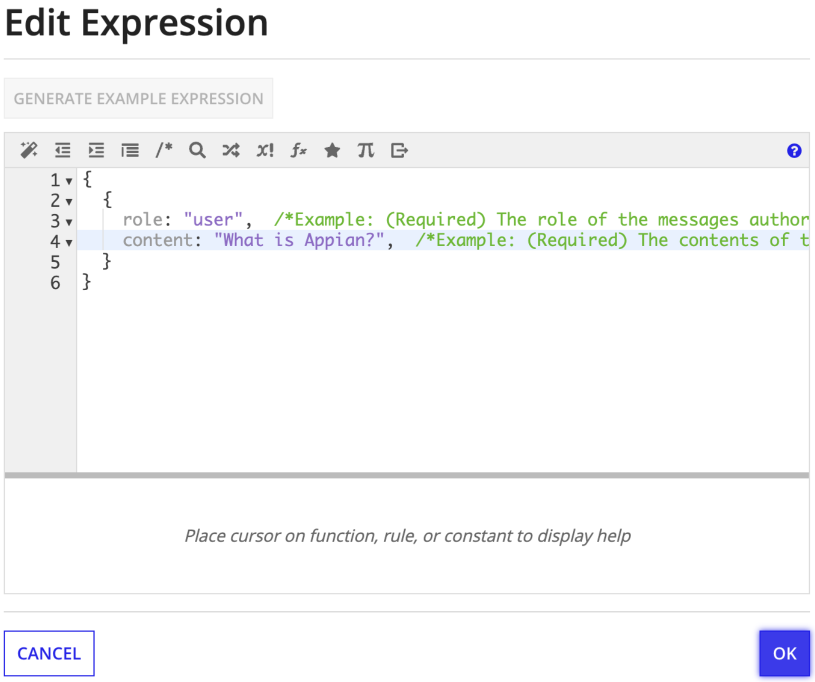
Integration	Function
Chat Completion (ChatGPT)	Given a chat conversation, the model will return a chat completion response.
Create Completion	Creates a completion for the provided prompt and parameters. Use Function Calling to have GPT select a function to call and provide arguments to call that function. (Soon to be deprecated, use chat completions instead)
Create Completion Edit	Given a prompt and an instruction, the model will return an edited version of the prompt. (Soon to be deprecated, use chat completions intead)
Create DALLE Image	Given a prompt and/or an input image, the model will generate a new image.
Transcribe Audio	Transcribes audio into the input language.
Create Translation	Translates audio into English.
Edit/Extend DALLE Image	Creates an edited or extended image given an original image and a prompt.
Create Variation of Image	Creates a variation of a given image
Create Vector Embedding	Creates an embedding vector representing the input text.
Upload a document	Upload a file that contains document(s) to be used across various endpoints/features. Currently, the size of all the files uploaded by one organization can be up to 1 GB. Please contact us if you need to increase the storage limit.
Return OpenAI Files	Returns a list of files that belong to the user's organization.
Delete OpenAI File	Delete a file
Return OpenAI File Information	Returns information about a specific file.
Return OpenAI File Contents	Returns the contents of the specified file
Create Fine	Tuning Job - Creates a job that fine-tunes a specified model from a given dataset. Response includes details of the enqueued job including job status and the name of the fine-tuned models once complete.
List Fine	Tuning Jobs - List your organization's fine-tuning jobs.
Cancel Fine	Tuning Job - Immediately cancel a fine-tune job.
Retrieve Fine	Tune Job Info - Gets info about the fine-tune job.
Get Fine	Tuning Job Status - Get fine-grained status updates for a fine-tune job.
List Available Models	Lists the currently available models and provides basic information about each one such as the owner and availability.
Delete Fine	Tuning Model - Delete a fine-tuned model. You must have the Owner role in your organization.
Retrieve Model Instance Information	Retrieves a model instance, providing basic information about the model such as the owner and permissioning.
Content Policy Violation	Classifies if text violates OpenAI's Content Policy

Create JSON Lines File	Creates a JSON Lines File from Appian data. This is the file format OpenAI expects to receive for fine-tuning jobs. Use this generated file to upload and fine-tune a model.

Basic ChatGPT Chatting Interaction

This will walk you through basic chatting functionality with ChatGPT using the /chat/completions endpoint.

1. Set up OpenAI account and OpenAI connected system (instructions below)
2. Select whether the integration reads or writes data from the initial dropdown.
    a. (Reads Data) – Queries data from the API and allows the result to be stored as a local variable.
    b. (Modifies Data) – Mutates data from the API and requires an onSuccess and onError fields to handle the results of the call. Modifying data requires the user to interact with Appian in some way to trigger the request.
3. Select "creates a completion for the chat message"
4. Fill out the required fields
    a. It is highly recommended that you use the "Specify values for each input" UI to fill in your field values and "Define all values with a single expression" to gain insight into the descriptions of all the available parameters.
5. For a basic chat interaction, the only parameters required are "model" and "messages" with at least one {role: "", content: ""} object. The "Generate Expression" button will allow you to introspect all possible fields that you can fill out, but these are the only required fields for this interaction. For example, messages can be filled out as follows. As the interaction with gpt grows, pass in more of these objects, listing the role as assistant to record GPT's responses.



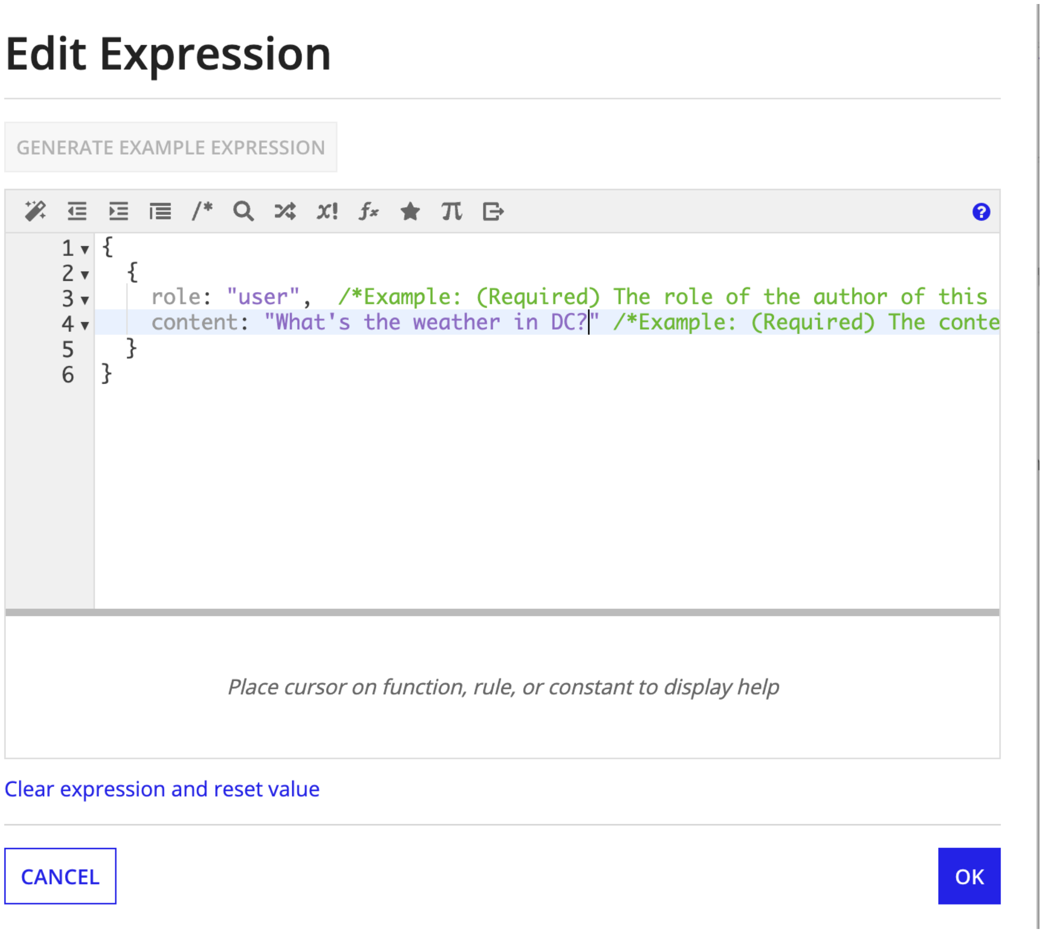
```
{
  {
    role: "user",  /*Example: (Required) The role of the messages author
    content: "What is Appian?",  /*Example: (Required) The contents of t
  }
}
```

Place cursor on function, rule, or constant to display help

ChatGPT Function Calling

Function calling is a way to allow GPT to select relevant functions to call to ground its answer in user data. The developer lists descriptions of functions along with a user query and GPT will be able to respond with the correct function to call and the arguments to call it with. The developer then can call this function and pass the function's response along with the original user's query so that GPT can answer the query based on the data provided. Read about function calling here and here before continuing.

1. Follow the instructions for "Basic GPT Interaction" above
2. Following the example provided from the OpenAI documentation, format your initial function parameter for the initial function calling query as follows. Make sure to wrap "parameters" field in a!toJson({})


Edit Expression
GENERATE EXAMPLE EXPRESSION
Place cursor on function, rule, or constant to display help
Clear expression and reset value
CANCEL
OK

```
{
  {
    role: "user",   /*Example: (Required) The role of the author of this
    content: "What's the weather in DC?" /*Example: (Required) The conte
  }
}
```

3.

Unset

```
{
  {
    name: "get_current_weather",
    /*Example: (Required) The name of the function to be called. Must be a-z,
A-Z, 0-9, or contain underscores and dashes, with a maximum length of 64.*/
```

```
    description: "Get the current weather in a given location",
    /*Example: A description of what the function does, used by the model to
choose when and how to call the function.*/
    parameters: a!toJson(
      {
        type: "object",
        properties: {
          location: {
            type: "string",
            description: "The city and state, e.g. San Francisco, CA"
          },
          unit: {
            type: "string",
            enum: { "celsius", "fahrenheit" }
          }
        },
        required: { "location" }
      }
    )/*Example: (Required) The value for 'parameters' is dynamic and must be
wrapped in a!toJson. Example parameters value: a!toJson( { type: "object",
properties: { location: { type: "string", description: "The city and state,
e.g. San Francisco, CA" }, unit: { type: "string", enum: { "celsius",
"fahrenheit" } } }, required: { "location" } }). The parameters the functions
accepts, described as a JSON Schema object. See the
[guide](/docs/guides/gpt/function-calling) for examples, and the [JSON Schema
reference](https://json-schema.org/understanding-json-schema/) for
documentation about the format.To describe a function that accepts no
parameters, provide the value `{"type": "object", "properties": {}}`.*/

  }
}
```

   a. The response will include the function it chose to call (in other examples you can list more than one function) along with the arguments to call it. With this response, call the function of choice (in this example, it would most likely be another integration to get location weather, but this could also be used to call other SAIL functions)

4. After receiving the response from the called function, you can pass this information back to GPT to answer the user's initial query
   a. The functions parameter will remain the same, while the messages parameter will look like:

Unset

```
{
  {
    {
      role: "user",
      content: "What is the weather like in Boston?"
    },
    {
      role: "assistant",
      content: null,
      function_call: {
        name: "get_current_weather",
        arguments: a!toJson({ location: "Boston, MA" })
      }
    },
    {
      role: "function",
      name: "get_current_weather",
      content: {
        temperature: "22",
        unit: "celsius",
        description: "Sunny"
      }
    }
  }
}
```

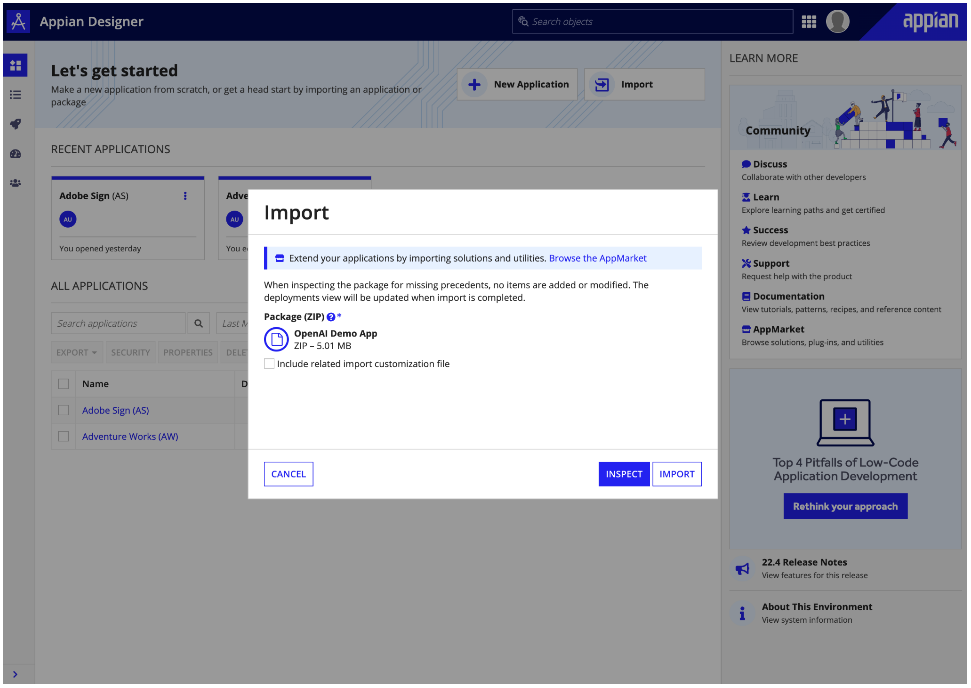
   b. Optionally use the “function_call” parameter which controls how the model responds to function calls. "none" means the model does not call a function, and responds to the end-user. "auto" means the model can pick between an end-user or calling a function. Specifying a particular function via `{"name":\ "my_function"}` forces the model to call that function. "none" is the default when no functions are present. "auto" is the default if functions are present.

5. The final response is GPT responding to the user’s initial query using the data provided from the function.

OpenAI Sample App Setup

This will walk you through importing the sample application and loading in a SQL file into the cloud database. The sample app has not been updated to reflect the switch to the /chat/completions endpoint. Make sure to use /chat/completions when building your own application.

1. Import the OpenAIDemoApp.zip file into your Appian environment.


Appian Designer
Search objects
appian
Let's get started
Make a new application from scratch, or get a head start by importing an application or package
New Application
Import
RECENT APPLICATIONS
Adobe Sign (AS)
AU
You opened yesterday
ALL APPLICATIONS
Search applications
EXPORT
SECURITY
PROPERTIES
Name
Adobe Sign (AS)
Adventure Works (AW)
LEARN MORE
Community
Discuss
Collaborate with other developers
Learn
Explore learning paths and get certified
Success
Review development best practices
Support
Request help with the product
Documentation
View tutorials, patterns, recipes, and reference content
AppMarket
Browse solutions, plug-ins, and utilities
Top 4 Pitfalls of Low-Code Application Development
Rethink your approach
22.4 Release Notes
View features for this release
About This Environment
View system information

Import

Extend your applications by importing solutions and utilities. Browse the AppMarket

When inspecting the package for missing precedents, no items are added or modified. The deployments view will be updated when import is completed.

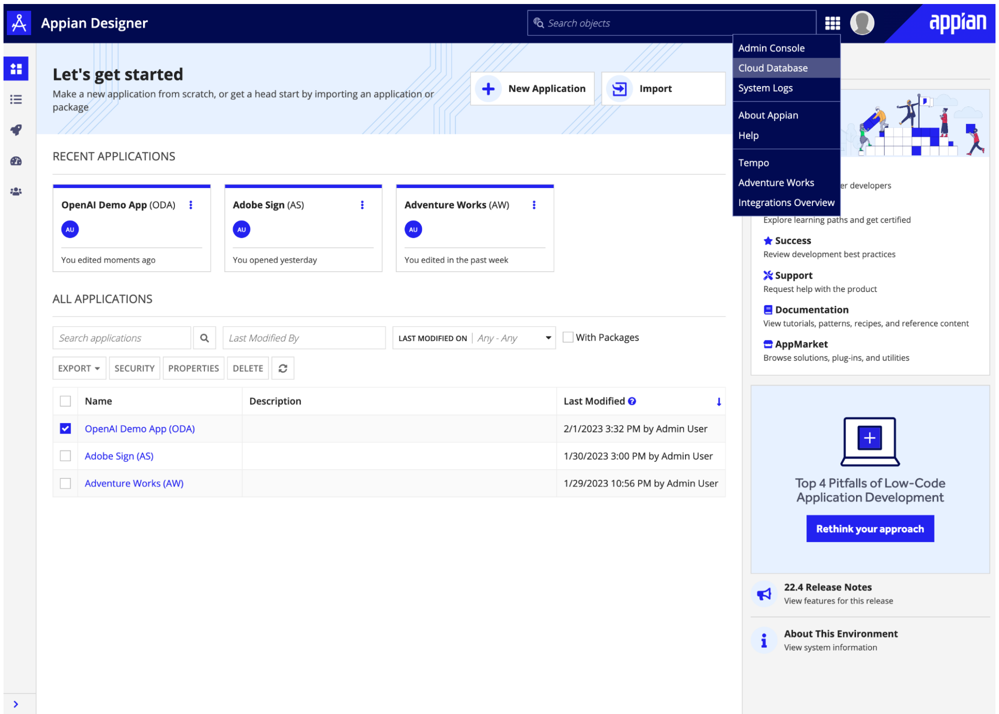
Package (ZIP) *

OpenAI Demo App
ZIP – 5.01 MB

□ Include related import customization file

CANCEL INSPECT IMPORT

2. Click the waffle menu in the top right corner and open the “Cloud Database”



Let's get started

Make a new application from scratch, or get a head start by importing an application or package

RECENT APPLICATIONS

OpenAI Demo App (ODA) — You edited moments ago

Adobe Sign (AS) — You opened yesterday

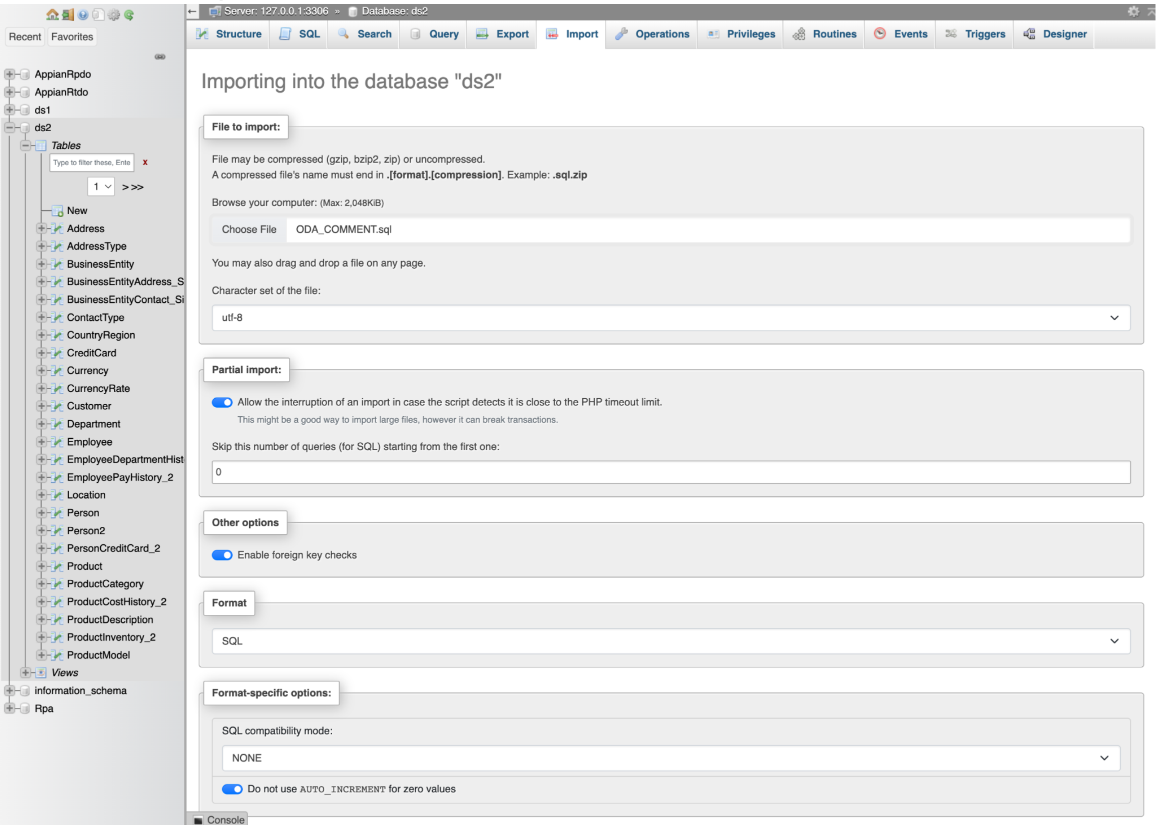
Adventure Works (AW) — You edited in the past week

ALL APPLICATIONS

Search applications | Last Modified By | LAST MODIFIED ON Any - Any

	Name	Description	Last Modified
☑	OpenAI Demo App (ODA)		2/1/2023 3:32 PM by Admin User
□	Adobe Sign (AS)		1/30/2023 3:00 PM by Admin User
□	Adventure Works (AW)		1/29/2023 10:56 PM by Admin User

3. Click "Import" and load the attached SQL file



Importing into the database "ds2"

File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in **.[format].[compression]**. Example: **.sql.zip**

Browse your computer: (Max: 2,048KiB)

Choose File ODA_COMMENT.sql

You may also drag and drop a file on any page.

Character set of the file: utf-8

Partial import:

● Allow the interruption of an import in case the script detects it is close to the PHP timeout limit.
This might be a good way to import large files, however it can break transactions.

Skip this number of queries (for SQL) starting from the first one: 0

Other options

● Enable foreign key checks

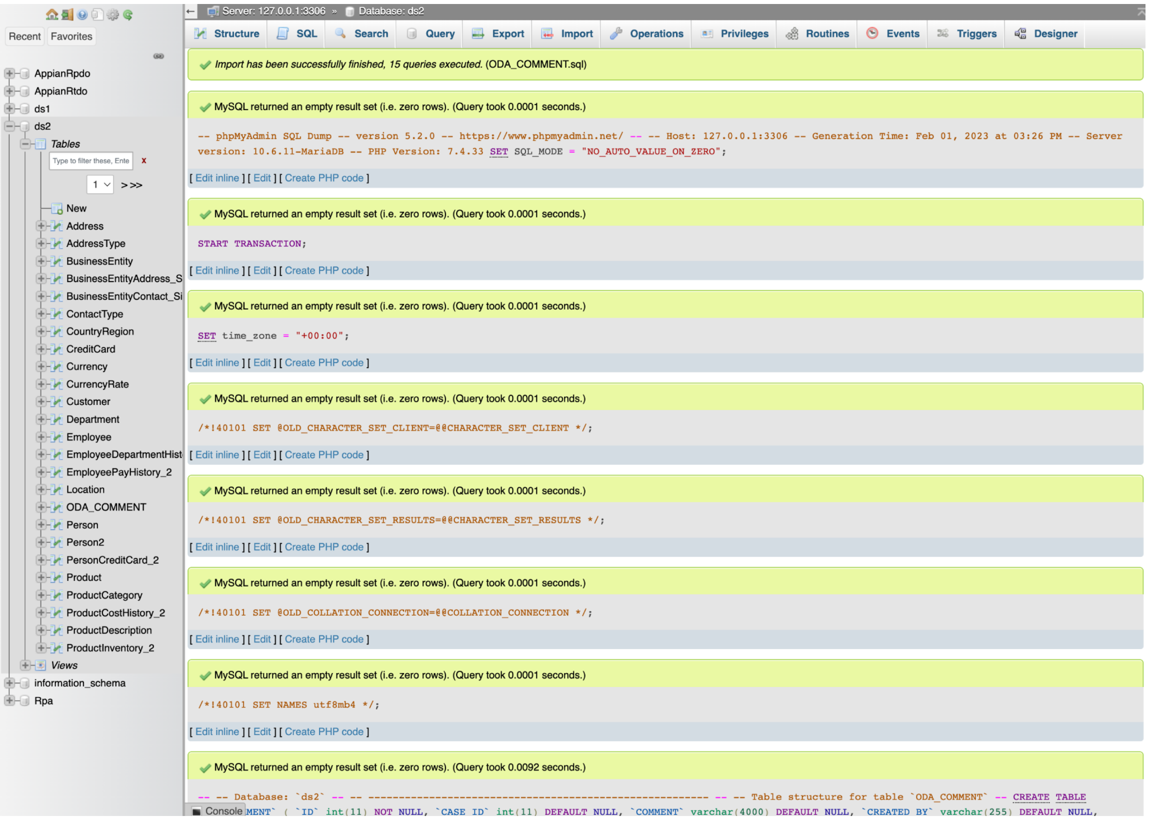
Format

SQL

Format-specific options:

SQL compatibility mode: NONE

● Do not use AUTO_INCREMENT for zero values



✓ *Import has been successfully finished, 15 queries executed.* (ODA_COMMENT.sql)

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0001 seconds.)

```
-- phpMyAdmin SQL Dump -- version 5.2.0 -- https://www.phpmyadmin.net/ -- -- Host: 127.0.0.1:3306 -- Generation Time: Feb 01, 2023 at 03:26 PM -- Server
version: 10.6.11-MariaDB -- PHP Version: 7.4.33 SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
```

[Edit inline] [Edit] [Create PHP code]

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0001 seconds.)

```
START TRANSACTION;
```

[Edit inline] [Edit] [Create PHP code]

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0001 seconds.)

```
SET time_zone = "+00:00";
```

[Edit inline] [Edit] [Create PHP code]

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0001 seconds.)

```
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
```

[Edit inline] [Edit] [Create PHP code]

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0001 seconds.)

```
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
```

[Edit inline] [Edit] [Create PHP code]

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0001 seconds.)

```
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
```

[Edit inline] [Edit] [Create PHP code]

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0001 seconds.)

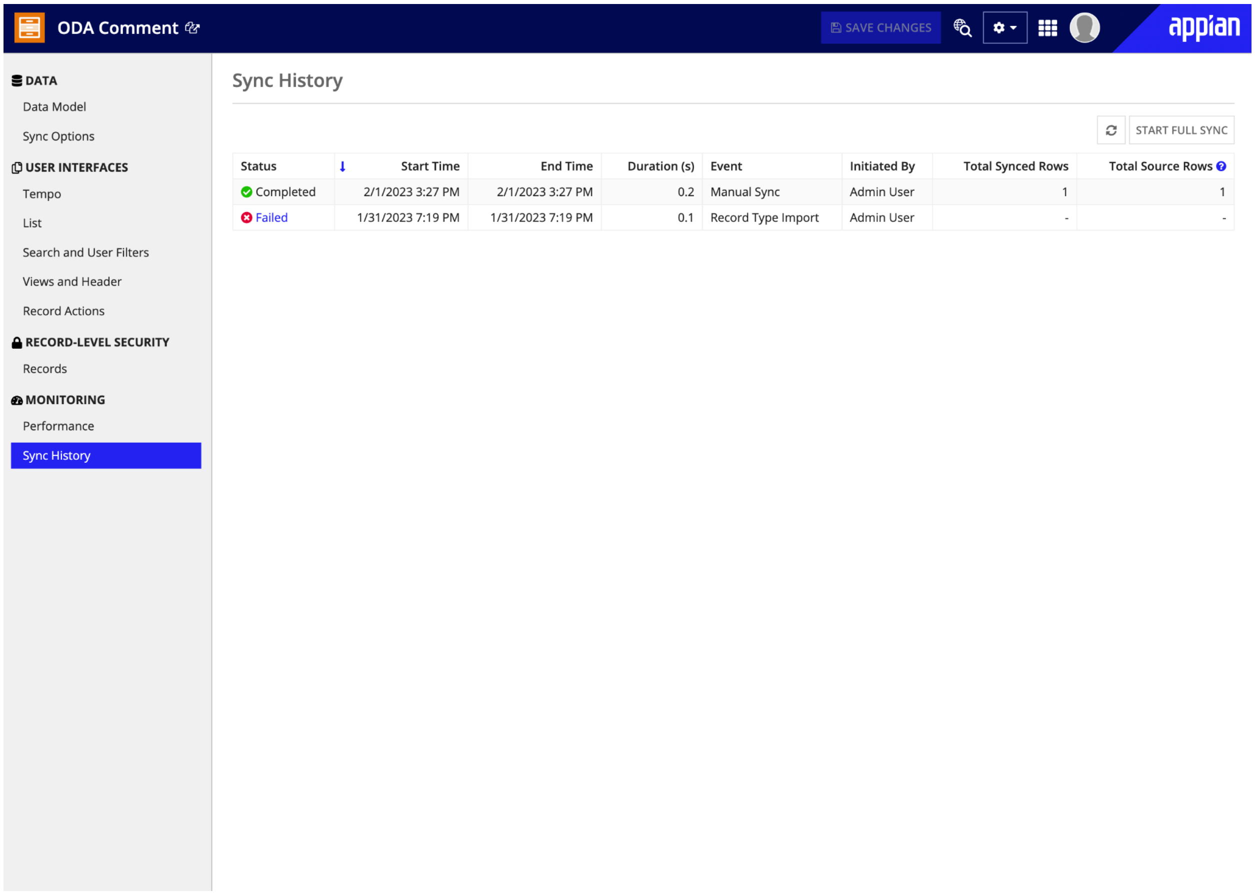
```
/*!40101 SET NAMES utf8mb4 */;
```

[Edit inline] [Edit] [Create PHP code]

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0092 seconds.)

```
-- -- Database: `ds2` -- -- -------------------------------------------------------- -- -- Table structure for table `ODA_COMMENT` -- CREATE TABLE
MENT` ( `ID` int(11) NOT NULL, `CASE ID` int(11) DEFAULT NULL, `COMMENT` varchar(4000) DEFAULT NULL, `CREATED BY` varchar(255) DEFAULT NULL,
```

4. In Designer, navigate to the Sync History of ODA Comment and click “Start Full Sync.”



Sync History

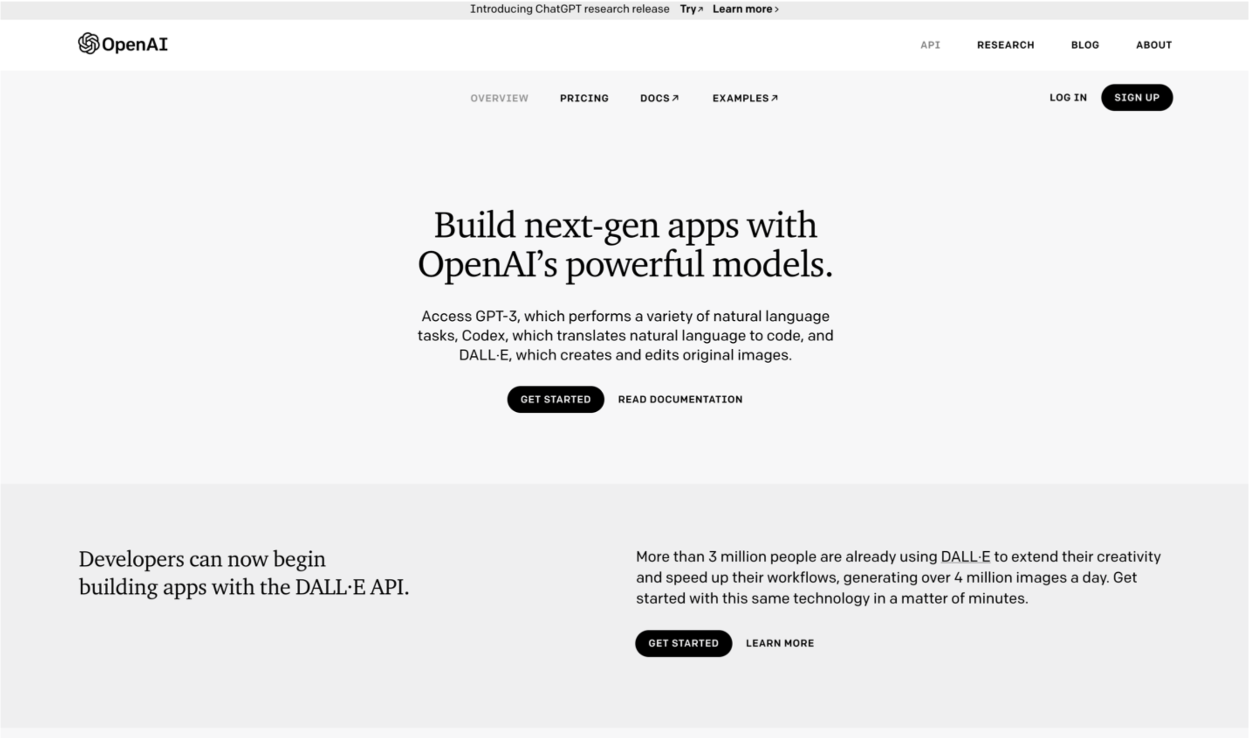
Status	Start Time	End Time	Duration (s)	Event	Initiated By	Total Synced Rows	Total Source Rows
✓ Completed	2/1/2023 3:27 PM	2/1/2023 3:27 PM	0.2	Manual Sync	Admin User	1	1
✗ Failed	1/31/2023 7:19 PM	1/31/2023 7:19 PM	0.1	Record Type Import	Admin User	-	-

5. Navigate to ODA integrationsOverview site, click the site link, and test out OpenAI's summarization capabilities on the example case management interface.

OpenAI Auth Setup

This will walk you through creating an OpenAI account and accessing the necessary credentials to use the connected system.

1. Navigate to OpenAI's API docs and sign up for an account by clicking "Get Started".



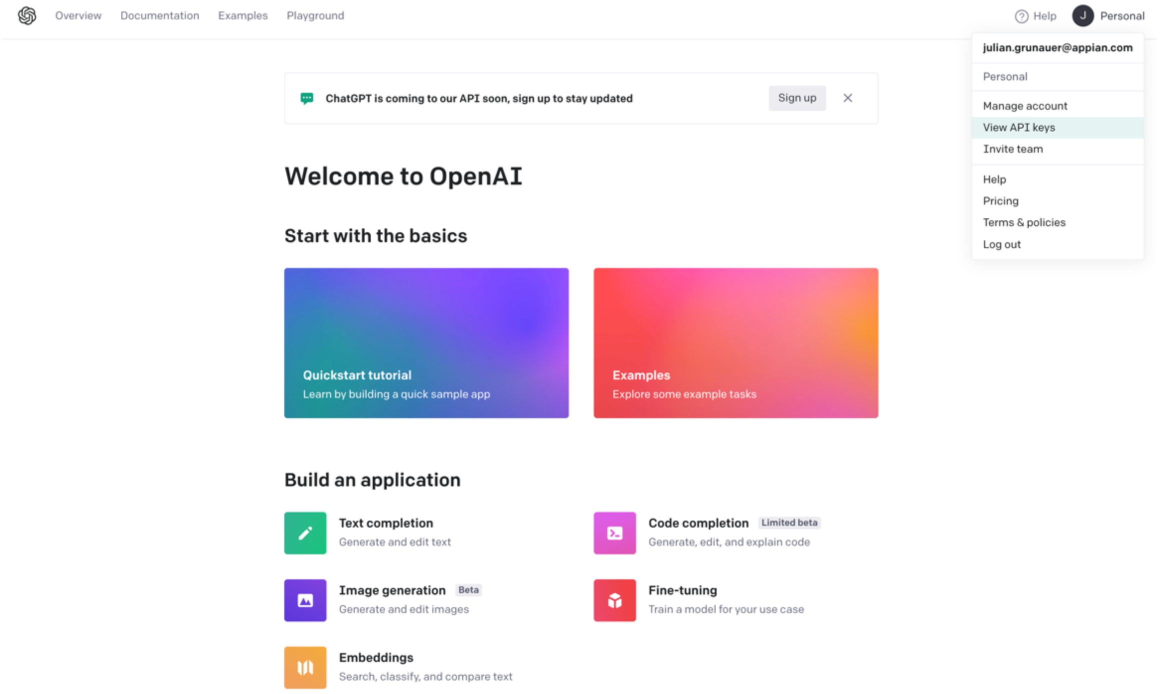
Build next-gen apps with OpenAI's powerful models.

Access GPT-3, which performs a variety of natural language tasks, Codex, which translates natural language to code, and DALL·E, which creates and edits original images.

Developers can now begin building apps with the DALL·E API.

More than 3 million people are already using DALL·E to extend their creativity and speed up their workflows, generating over 4 million images a day. Get started with this same technology in a matter of minutes.

2. Click on "Personal" then "View API keys".



ChatGPT is coming to our API soon, sign up to stay updated

Welcome to OpenAI

Start with the basics

Quickstart tutorial
Learn by building a quick sample app

Examples
Explore some example tasks

Build an application

Text completion
Generate and edit text

Code completion Limited beta
Generate, edit, and explain code

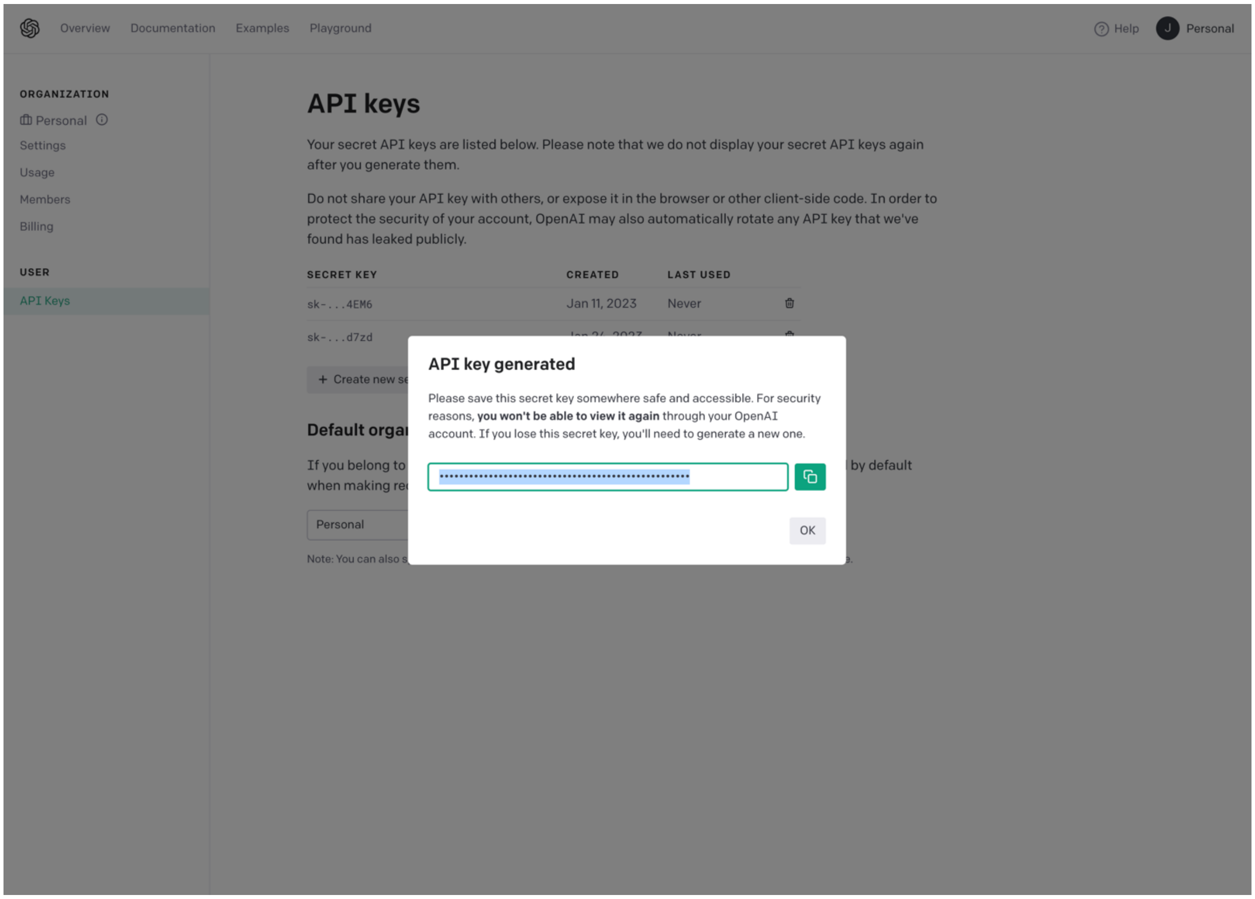
Image generation Beta
Generate and edit images

Fine-tuning
Train a model for your use case

Embeddings
Search, classify, and compare text

3. Click “Create new API key” then copy the generated API key.

Important Note: Make sure to copy this key as it will only be shown once.



API keys

Your secret API keys are listed below. Please note that we do not display your secret API keys again after you generate them.

Do not share your API key with others, or expose it in the browser or other client-side code. In order to protect the security of your account, OpenAI may also automatically rotate any API key that we've found has leaked publicly.

SECRET KEY	CREATED	LAST USED
sk-...4EM6	Jan 11, 2023	Never
sk-...d7zd	Jan 24, 2023	Never

Default orga

If you belong to … by default when making re

Personal

Note: You can also s … e.

API key generated

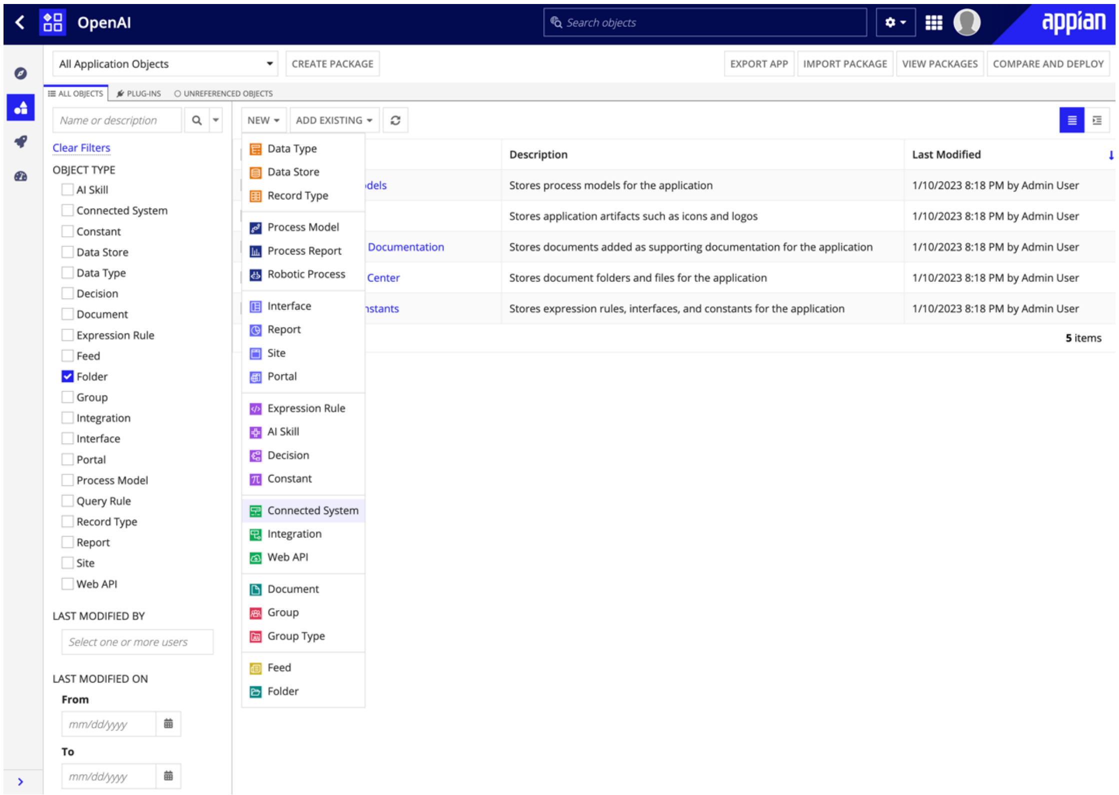
Please save this secret key somewhere safe and accessible. For security reasons, **you won't be able to view it again** through your OpenAI account. If you lose this secret key, you'll need to generate a new one.

Appian Connected System Setup

This will walk you through how to set up the connected system within your Appian instance.

1. Download the plugin and load it into your Appian Environment.

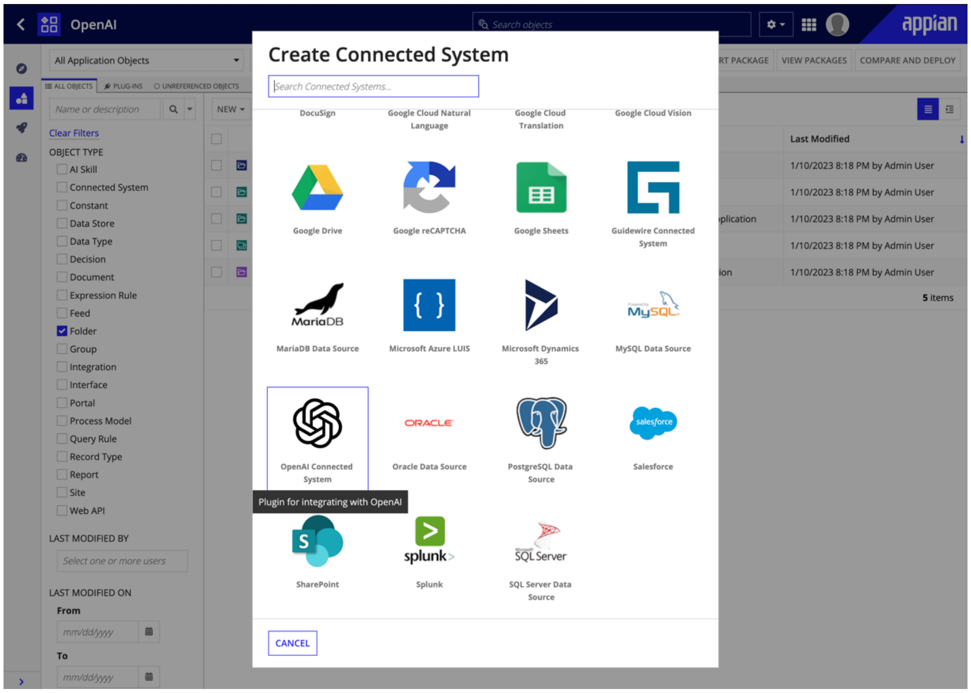
2. Click "New" then "Connected System".



	Description	Last Modified
dels	Stores process models for the application	1/10/2023 8:18 PM by Admin User
	Stores application artifacts such as icons and logos	1/10/2023 8:18 PM by Admin User
Documentation	Stores documents added as supporting documentation for the application	1/10/2023 8:18 PM by Admin User
Center	Stores document folders and files for the application	1/10/2023 8:18 PM by Admin User
nstants	Stores expression rules, interfaces, and constants for the application	1/10/2023 8:18 PM by Admin User

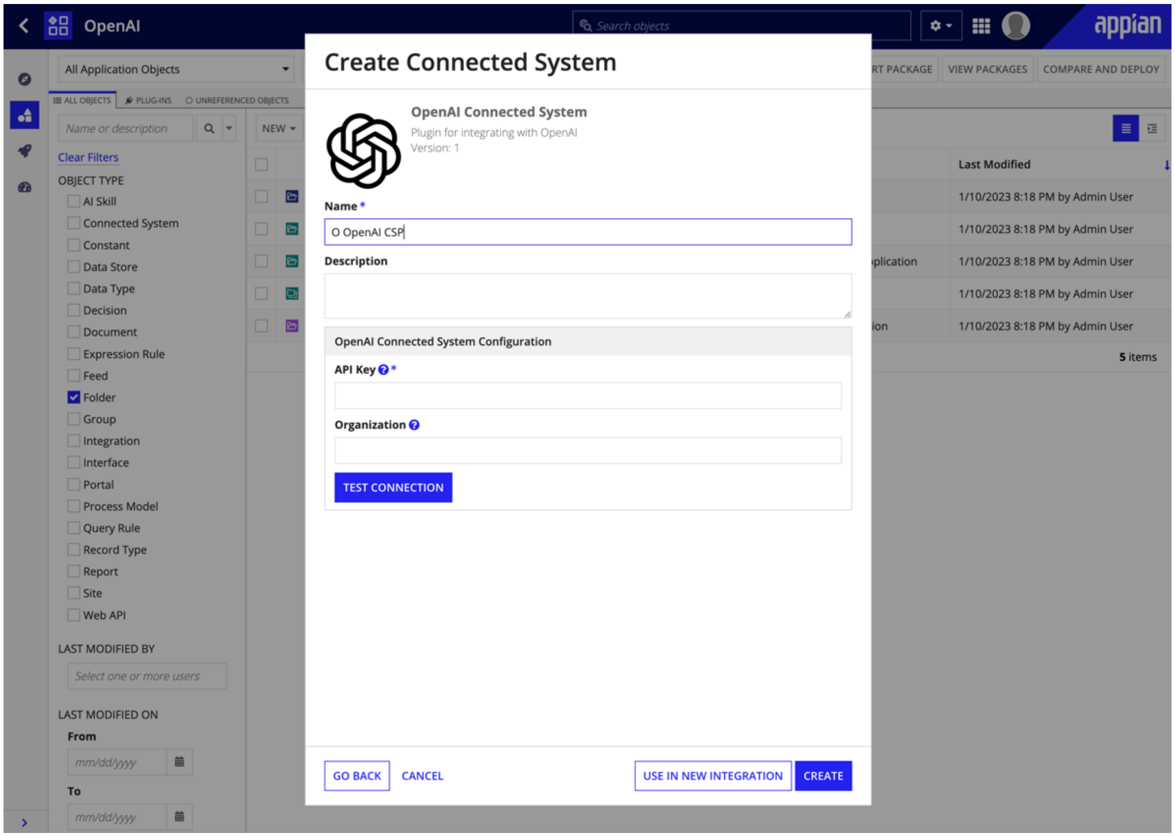
5 items

3. Click "OpenAI Connected System".



4. Name, describe, and input your API key into the Connected System. The organization parameter is option:
   - "For users who belong to multiple organizations, you can pass a header to specify which organization is used for an API request. Usage from these API requests will count against the specified organization's subscription quota."

5. Press "Test Connection" to validate that the authentication is properly configured.



1/10/2023 8:18 PM by Admin User
1/10/2023 8:18 PM by Admin User
1/10/2023 8:18 PM by Admin User
1/10/2023 8:18 PM by Admin User
1/10/2023 8:18 PM by Admin User

Create Connected System

OpenAI Connected System
Plugin for integrating with OpenAI
Version: 1

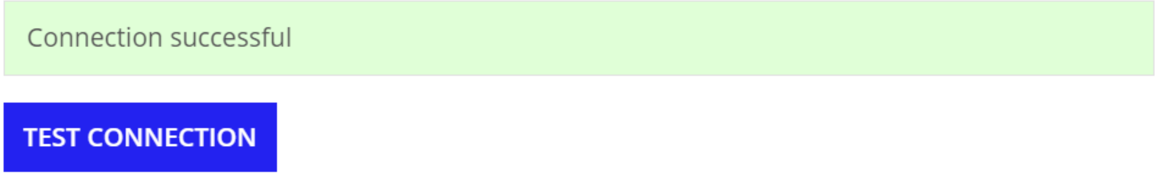
Name *
O OpenAI CSP

Description

OpenAI Connected System Configuration

API Key *

Organization



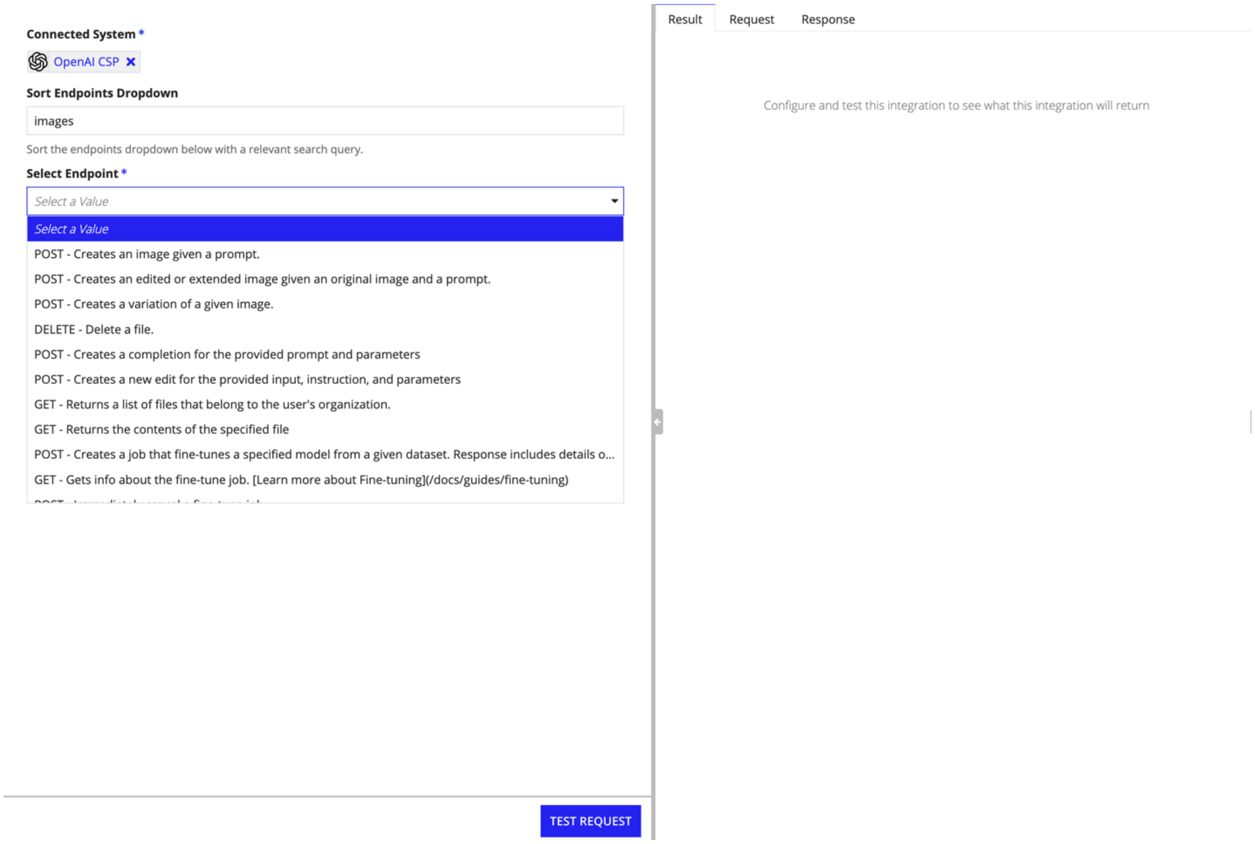
Connection successful

Integration Configuration

This section provides information about how to set up your integrations with the connected system.

1. Select whether the integration reads or writes data from the initial dropdown.
   - (Reads Data) – Queries data from the API and allows the result to be stored as a local variable.
   - (Modifies Data) – Mutates data from the API and requires a onSuccess and onError fields to handle the results of the call. Modifying data requires the user to interact with Appian in some way to trigger the request.

2. Select an endpoint from the dropdown. To help with selection, there is a search bar that will sort the dropdown list based on user input. For example, if a user searches for images, the endpoints relating to image generation will appear at the top of the dropdown list.



Connected System *

OpenAI CSP

Sort Endpoints Dropdown

images

Sort the endpoints dropdown below with a relevant search query.

Select Endpoint *

Select a Value

Select a Value

POST - Creates an image given a prompt.

POST - Creates an edited or extended image given an original image and a prompt.

POST - Creates a variation of a given image.

DELETE - Delete a file.

POST - Creates a completion for the provided prompt and parameters

POST - Creates a new edit for the provided input, instruction, and parameters

GET - Returns a list of files that belong to the user's organization.

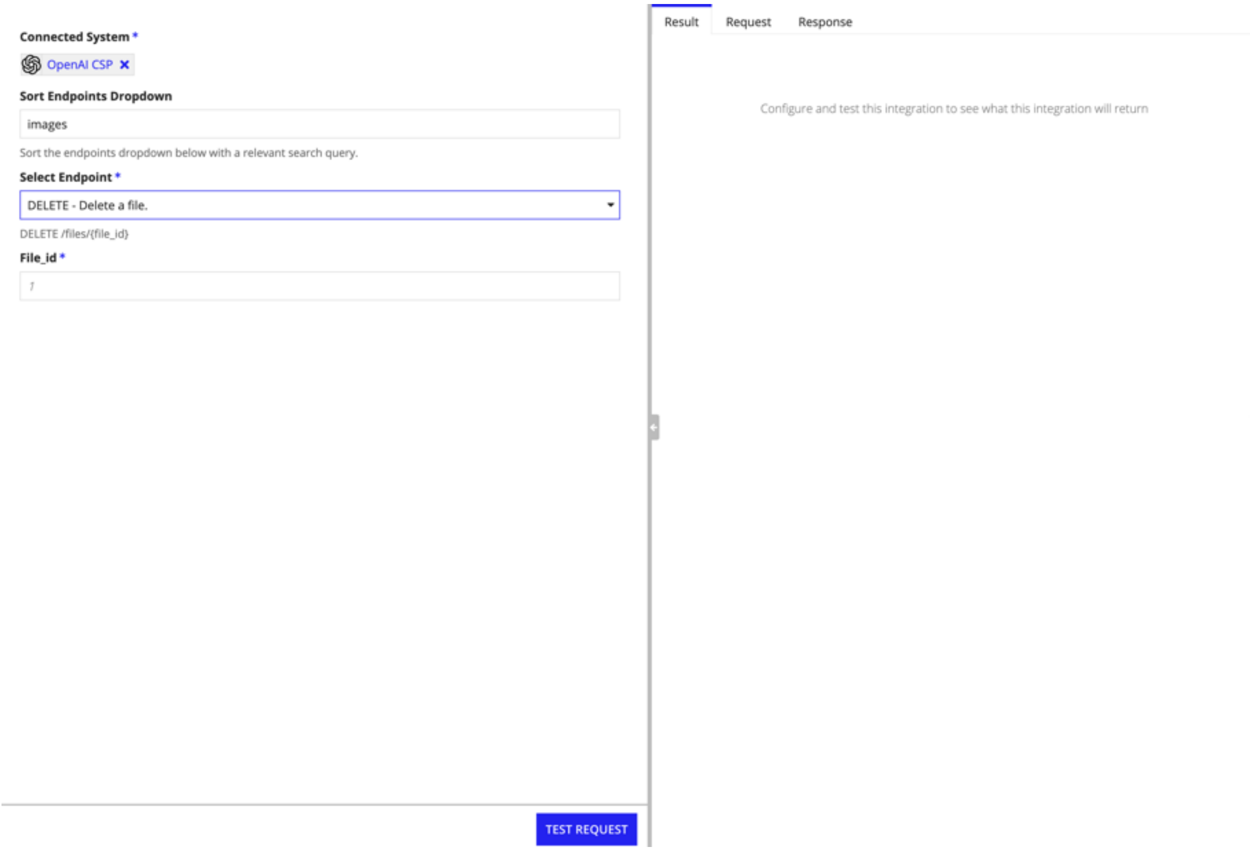
GET - Returns the contents of the specified file

POST - Creates a job that fine-tunes a specified model from a given dataset. Response includes details o...

GET - Gets info about the fine-tune job. [Learn more about Fine-tuning](/docs/guides/fine-tuning)

Configure and test this integration to see what this integration will return

3. Some endpoints have required path parameters. These fields are automatically generated and added to the url for you. For example, `DELETE /files/{file_id}`



Connected System *

OpenAI CSP

Sort Endpoints Dropdown

images

Sort the endpoints dropdown below with a relevant search query.

Select Endpoint *

DELETE - Delete a file.

DELETE /files/{file_id}

File_id *

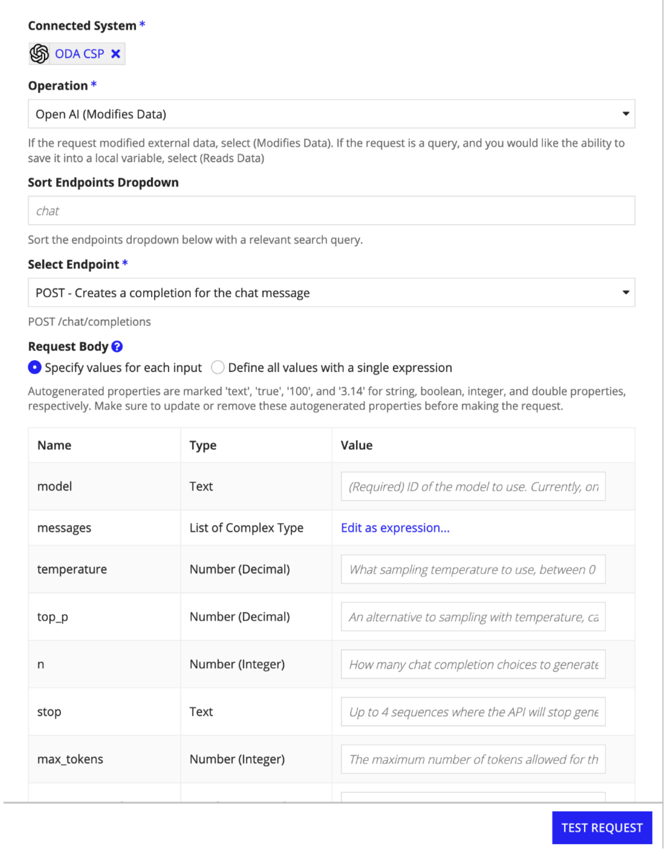
Configure and test this integration to see what this integration will return

Integration Information

This section contains key information necessary for successful utilization of the integrations associated with this connected system.

Most POST requests require a request body. Appian provides two interfaces for working with complex requests—”Specify values for each input” or “Define all values with a single expression.”

- Specify values for each input: allows for users to specify only the properties they need; any properties left blank will not be sent in the request.
- Define all values with a single expression: autogenerates an example expression for the entire request.



Connected System *

ODA CSP ✕

Operation *

Open AI (Modifies Data)

If the request modified external data, select (Modifies Data). If the request is a query, and you would like the ability to save it into a local variable, select (Reads Data)

Sort Endpoints Dropdown

chat

Sort the endpoints dropdown below with a relevant search query.

Select Endpoint *

POST - Creates a completion for the chat message

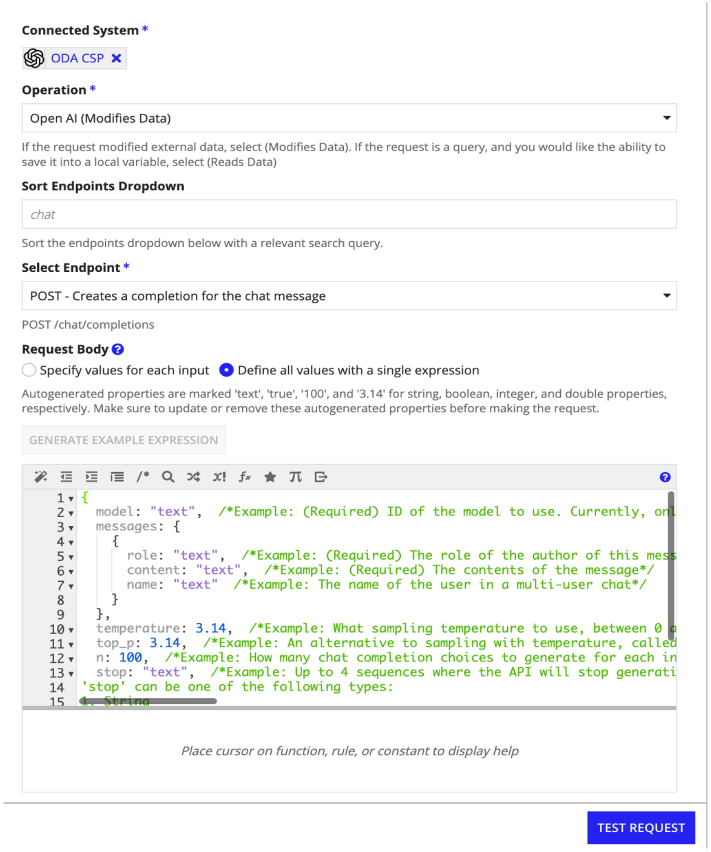
POST /chat/completions

Request Body

● Specify values for each input ○ Define all values with a single expression

Autogenerated properties are marked 'text', 'true', '100', and '3.14' for string, boolean, integer, and double properties, respectively. Make sure to update or remove these autogenerated properties before making the request.

Name	Type	Value
model	Text	*(Required) ID of the model to use. Currently, on*
messages	List of Complex Type	Edit as expression...
temperature	Number (Decimal)	*What sampling temperature to use, between 0*
top_p	Number (Decimal)	*An alternative to sampling with temperature, ca*
n	Number (Integer)	*How many chat completion choices to generate*
stop	Text	*Up to 4 sequences where the API will stop gene*
max_tokens	Number (Integer)	*The maximum number of tokens allowed for th*



Connected System *

ODA CSP ✕

Operation *

Open AI (Modifies Data)

If the request modified external data, select (Modifies Data). If the request is a query, and you would like the ability to save it into a local variable, select (Reads Data)

Sort Endpoints Dropdown

chat

Sort the endpoints dropdown below with a relevant search query.

Select Endpoint *

POST - Creates a completion for the chat message

POST /chat/completions

Request Body

○ Specify values for each input ● Define all values with a single expression

Autogenerated properties are marked 'text', 'true', '100', and '3.14' for string, boolean, integer, and double properties, respectively. Make sure to update or remove these autogenerated properties before making the request.

```
{
  model: "text",  /*Example: (Required) ID of the model to use. Currently, onl
  messages: {
    {
      role: "text",  /*Example: (Required) The role of the author of this mess
      content: "text",  /*Example: (Required) The contents of the message*/
      name: "text"  /*Example: The name of the user in a multi-user chat*/
    }
  },
  temperature: 3.14,  /*Example: What sampling temperature to use, between 0 a
  top_p: 3.14,  /*Example: An alternative to sampling with temperature, called
  n: 100,  /*Example: How many chat completion choices to generate for each in
  stop: "text",  /*Example: Up to 4 sequences where the API will stop generati
'stop' can be one of the following types:
1. String
```

Place cursor on function, rule, or constant to display help

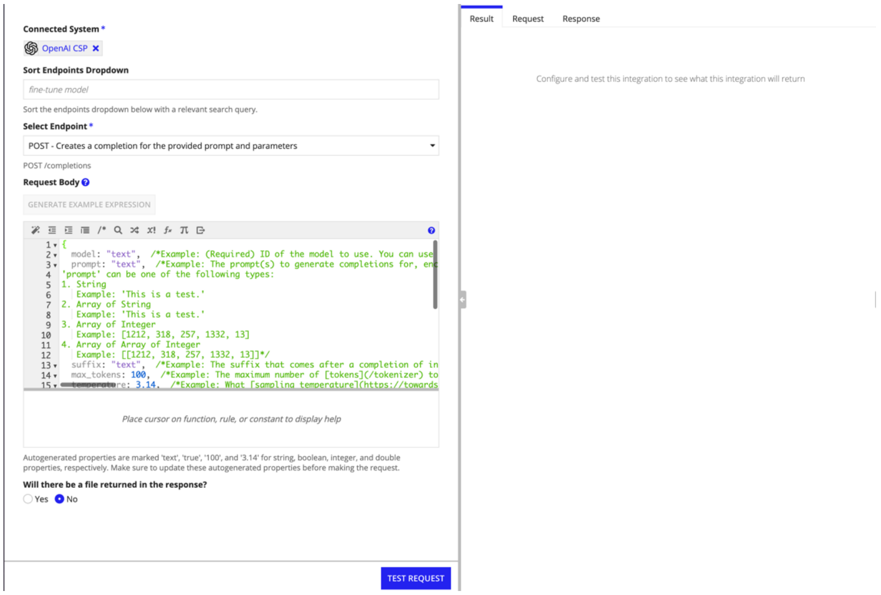
Recommended Usage

Click “Generate Example Expression” to autogenerate the properties required to make the request. This view allows you to see the totality of parameters available, nested required properties that may be hidden in the other view, as well as complete parameter descriptions. Unless you intend to use most/all of the fields, it is recommended that you use this view for reference. The two views will not overwrite each other if both are being edited. Whichever view is selected/saved at the time of execution will be the view used for the request.

Important Note: Remove or comment out superfluous keys/values.

Important Note: Only the autogenerated properties will be captured.
User inputted key/values that are not part of OpenAI's API specification will not be sent in the HTTP request. It is possible to use expression rules, record queries, or other forms of data manipulation to pass in values, as long as the keys/data structure remain the same as the autogenerated properties.

Important Note: Certain fields are marked as "(Required)." These fields must be inputted before making the request



Configure and test this integration to see what this integration will return

Connected System *

OpenAI CSP ✕

Sort Endpoints Dropdown

fine-tune model

Sort the endpoints dropdown below with a relevant search query.

Select Endpoint *

POST - Creates a completion for the provided prompt and parameters

POST /completions

Request Body

```
{
  model: "text",  /*Example: (Required) ID of the model to use. You can use
  prompt: "text",  /*Example: The prompt(s) to generate completions for, enc
'prompt' can be one of the following types:
1. String
   Example: 'This is a test.'
2. Array of String
   Example: 'This is a test.'
3. Array of Integer
   Example: [1212, 318, 257, 1332, 13]
4. Array of Array of Integer
   Example: [[1212, 318, 257, 1332, 13]]*/
  suffix: "text",  /*Example: The suffix that comes after a completion of in
  max_tokens: 100,  /*Example: The maximum number of [tokens](/tokenizer) to
  temperature: 3.14,  /*Example: What [sampling temperature](https://towards
```

Place cursor on function, rule, or constant to display help

Autogenerated properties are marked 'text', 'true', '100', and '3.14' for string, boolean, integer, and double properties, respectively. Make sure to update these autogenerated properties before making the request.

Will there be a file returned in the response?

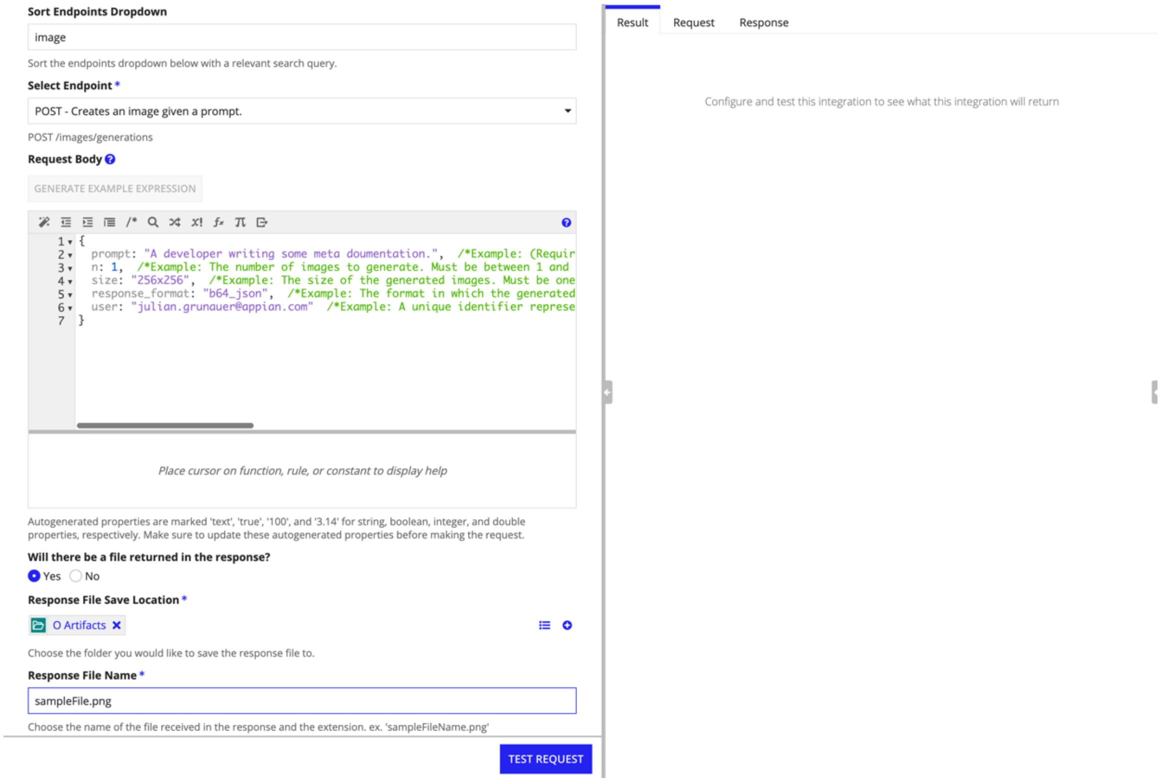
○ Yes ● No

Some requests have the option to receive a file back. If you expect to receive a file back, make sure to click "Yes" to the "Will there be a file returned in the response?" This will allow you to control the filename of the incoming document, as well as where to save this file in your Appian application.

Important Note: Make sure to add the extension of the file (ex. .png, .jpg, …) you expect to receive back.

Important Note: If you expect to receive multiple files back, they will be automatically indexed (ex. fileName1.png, fileName2.png, …)

Important Note: Currently the only endpoints returning files are those relating to image generation. Use the value "b64_json" for the "response_format" key and configure the filename and file location to catch the incoming file. Specify "url" to receive a url to the generated image.



Sort Endpoints Dropdown

image

Sort the endpoints dropdown below with a relevant search query.

Select Endpoint *

POST - Creates an image given a prompt.

POST /images/generations

Request Body

```
{
  prompt: "A developer writing some meta doumentation.",  /*Example: (Requir
  n: 1,  /*Example: The number of images to generate. Must be between 1 and
  size: "256x256",  /*Example: The size of the generated images. Must be one
  response_format: "b64_json",  /*Example: The format in which the generated
  user: "julian.grunauer@appian.com"  /*Example: A unique identifier represe
}
```

Place cursor on function, rule, or constant to display help

Autogenerated properties are marked 'text', 'true', '100', and '3.14' for string, boolean, integer, and double properties, respectively. Make sure to update these autogenerated properties before making the request.

Will there be a file returned in the response?

● Yes ○ No

Response File Save Location *

O Artifacts

Choose the folder you would like to save the response file to.

Response File Name *

sampleFile.png

Choose the name of the file received in the response and the extension. ex. 'sampleFileName.png'

TEST REQUEST
Result
Request
Response
Configure and test this integration to see what this integration will return

Pre-made models vary by usage. Make sure to use the correct model for the correct endpoints.

- For example, text-davinci-003, text-curie-001, text-babbage-001, text-ada-001, and custom, fine-tuned models can be used for the POST /completions endpoint, but only text-davinci-edit-001 can be used for the POST /edits endpoint. For embeddings, use the models listed here.
- Learn more about models and their usage here.

If you need to submit an empty string value, simply add a space between quotes (ex. `{prompt: “ “}`)

Fine-Tuning Flow

This section contains instructions on steps to train an OpenAI model based on Appian data.

Important Note. Fine-tuning is often not the best way to teach GPT about your information. Providing GPT with context along with your prompt often leads to the best results. Fine-tuning is not yet available for GPT-4.

Learn more from these links:
References

- Fine-Tuning vs Semantic Retrieval
- Walkthrough vector embeddings/context retrieval
- Large pdf example
- Multi-user chatbot
- Supabase documentation chatbot

1. Use the JSONLines operation to create a jsonLines file. Read the fine-tuning docs and text-completion guide. Make sure to append the filename with the extension ".jsonl"

Important Note. Use the extension ".jsonl" when setting the file name

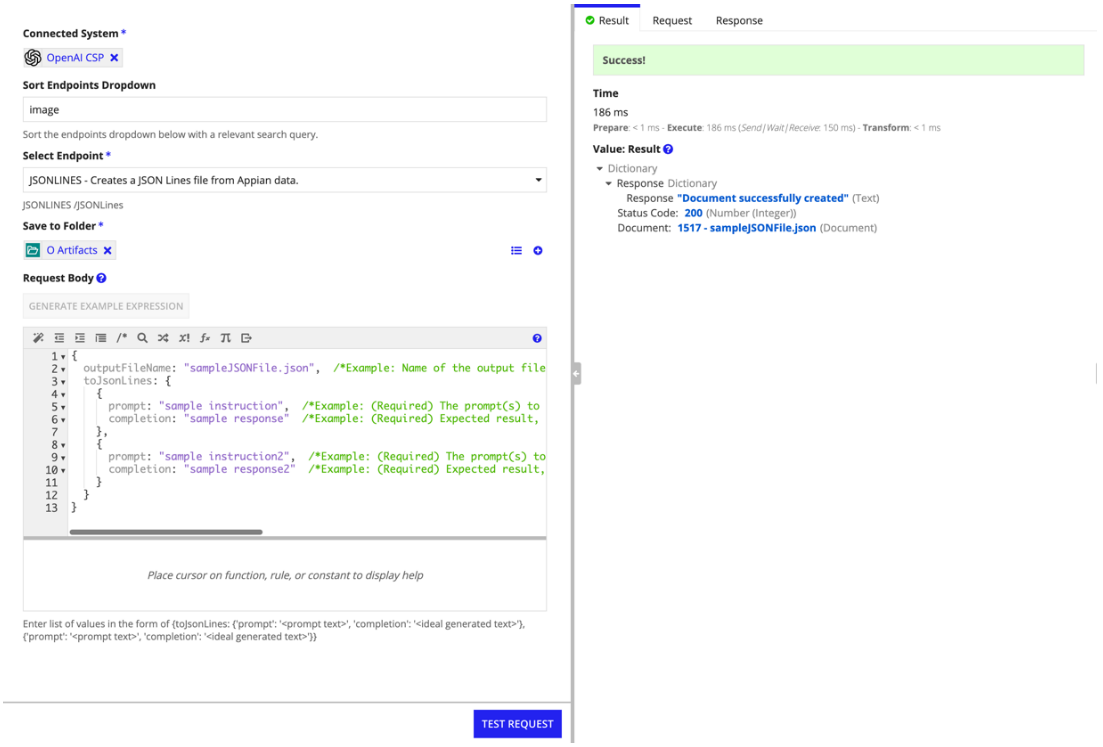
Important Note. You can either input the data directly into the expression box (as shown in the first example 1), or you can use a record with fields of type "prompt" and "completion" and pass in a record query as shown in example 2.

Sample code for expression query:

JavaScript

```
a!forEach(
  items: a!queryRecordType(
    recordType:
'recordType!{e2a3f34a-869c-45ed-9b10-27abf2a50c01}0
promptCompletion',
    fields: {
      'recordType!{e2a3f34a-869c-45ed-9b10-27abf2a50c01}0
promptCompletion.fields.{28b5e083-da90-4d94-bd7f-c840a998
53c8}prompt',
      'recordType!{e2a3f34a-869c-45ed-9b10-27abf2a50c01}0
promptCompletion.fields.{03686e9e-da88-476c-ac79-178d155d
7270}completion'
```

```
        },
              pagingInfo: a!pagingInfo(startIndex: 1,
          batchSize: 1000)
            ).data,
            expression: {
              prompt:
          fv!item['recordType!{e2a3f34a-869c-45ed-9b10-27ab
          f2a50c01}O
          promptCompletion.fields.{03686e9e-da88-476c-ac79-
          178d155d7270}completion'],
              completion:
          fv!item['recordType!{e2a3f34a-869c-45ed-9b10-27ab
          f2a50c01}O
```



Connected System *

OpenAI CSP ✕

Sort Endpoints Dropdown

image

Sort the endpoints dropdown below with a relevant search query.

Select Endpoint *

JSONLINES - Creates a JSON Lines file from Appian data.

JSONLINES /JSONLines

Save to Folder *

O Artifacts ✕

Request Body

GENERATE EXAMPLE EXPRESSION

```
{
  outputFileName: "sampleJSONFile.json",  /*Example: Name of the output file
  toJsonLines: {
    {
      prompt: "sample instruction",  /*Example: (Required) The prompt(s) to
      completion: "sample response"  /*Example: (Required) Expected result,
    },
    {
      prompt: "sample instruction2",  /*Example: (Required) The prompt(s) to
      completion: "sample response2"  /*Example: (Required) Expected result,
    }
  }
}
```

Place cursor on function, rule, or constant to display help

Enter list of values in the form of {toJsonLines: {'prompt': '<prompt text>', 'completion': '<ideal generated text>'}, {'prompt': '<prompt text>', 'completion': '<ideal generated text>'}}

TEST REQUEST

Success!

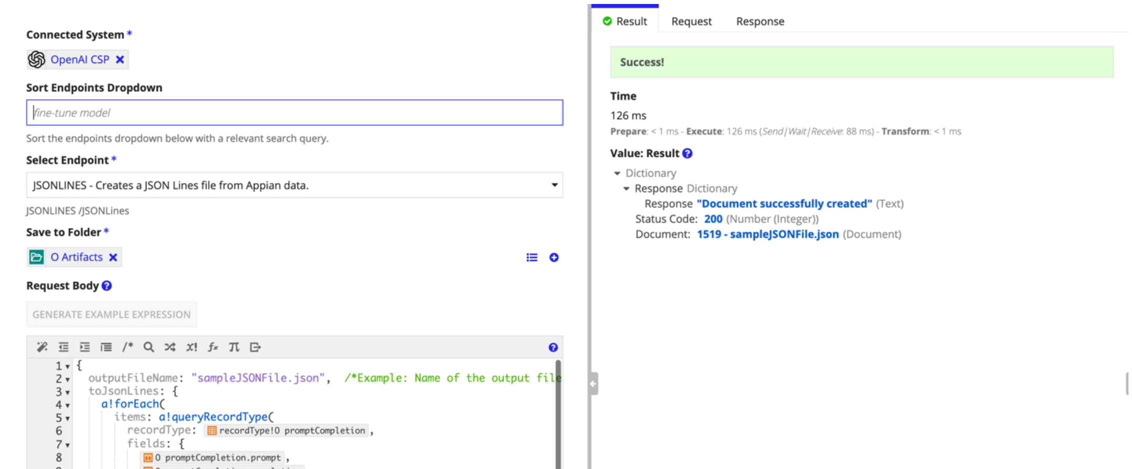
Time

186 ms

Prepare: < 1 ms - **Execute**: 186 ms (*Send | Wait | Receive*: 150 ms) - **Transform**: < 1 ms

Value: Result

```
▾ Dictionary
  ▾ Response Dictionary
      Response "Document successfully created" (Text)
    Status Code:  200 (Number (Integer))
    Document:  1517 - sampleJSONFile.json (Document)
```



Connected System *

OpenAI CSP ✕

Sort Endpoints Dropdown

fine-tune model

Sort the endpoints dropdown below with a relevant search query.

Select Endpoint *

JSONLINES - Creates a JSON Lines file from Appian data.

JSONLINES /JSONLines

Save to Folder *

O Artifacts ✕

Request Body

GENERATE EXAMPLE EXPRESSION

```
{
  outputFileName: "sampleJSONFile.json",  /*Example: Name of the output file
  toJsonLines: {
    a!forEach(
      items: a!queryRecordType(
        recordType: recordType!O promptCompletion ,
        fields: {
          O promptCompletion.prompt ,
```

Success!

Time

126 ms

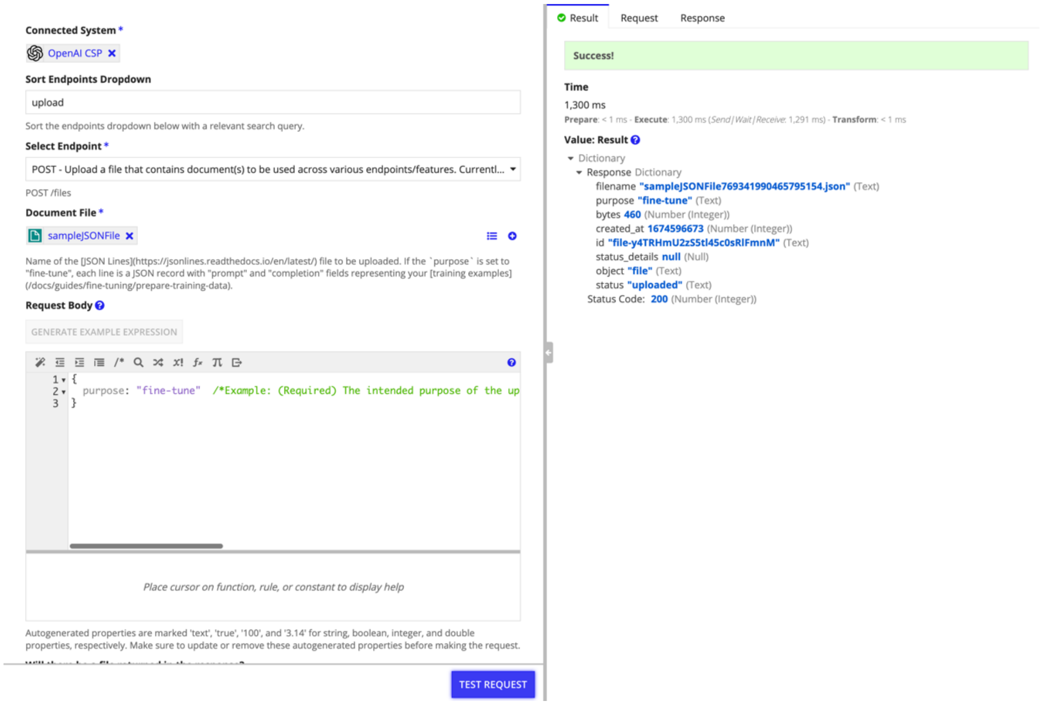
Prepare: < 1 ms - **Execute**: 126 ms (*Send | Wait | Receive*: 88 ms) - **Transform**: < 1 ms

Value: Result

```
▾ Dictionary
  ▾ Response Dictionary
      Response "Document successfully created" (Text)
    Status Code:  200 (Number (Integer))
    Document:  1519 - sampleJSONFile.json (Document)
```

2. Use the “Upload a file…” endpoint (POST /files) to upload the file to OpenAI.

Important Note: Make sure to set the purpose to “fine-tune” so that OpenAI knows that this file will be used to fine-tune a model.



Connected System *

OpenAI CSP

Sort Endpoints Dropdown

upload

Sort the endpoints dropdown below with a relevant search query.

Select Endpoint *

POST - Upload a file that contains document(s) to be used across various endpoints/features. Currentl...

POST /files

Document File *

sampleJSONFile

Name of the [JSON Lines](https://jsonlines.readthedocs.io/en/latest/) file to be uploaded. If the `purpose` is set to "fine-tune", each line is a JSON record with "prompt" and "completion" fields representing your [training examples](/docs/guides/fine-tuning/prepare-training-data).

Request Body

```
{
  purpose: "fine-tune"  /*Example: (Required) The intended purpose of the up
}
```

Place cursor on function, rule, or constant to display help

Autogenerated properties are marked 'text', 'true', '100', and '3.14' for string, boolean, integer, and double properties, respectively. Make sure to update or remove these autogenerated properties before making the request.

Success!

Time

1,300 ms

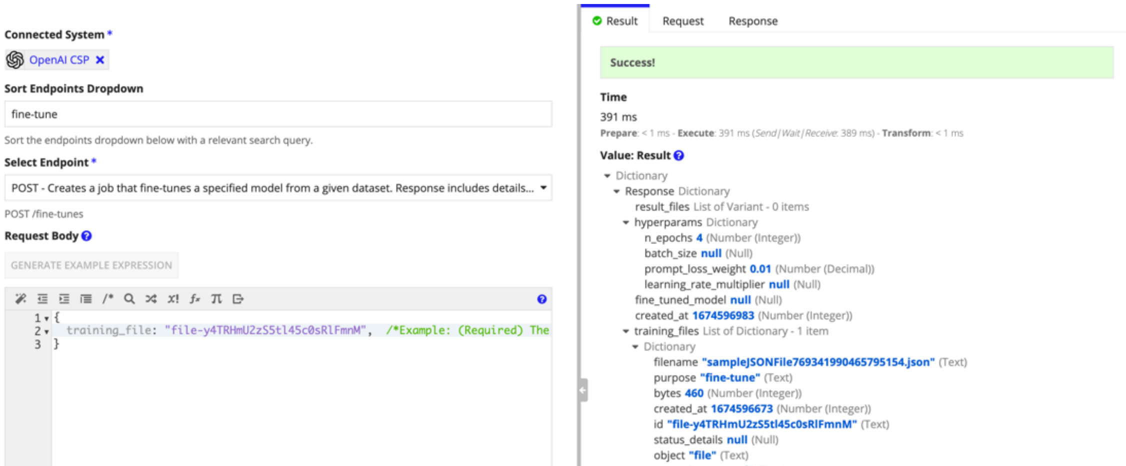
Prepare: < 1 ms - **Execute**: 1,300 ms (*Send / Wait / Receive*: 1,291 ms) - **Transform**: < 1 ms

Value: Result

```
▾ Dictionary
  ▾ Response Dictionary
      filename "sampleJSONFile7693419904657951 54.json" (Text)
      purpose "fine-tune" (Text)
      bytes 460 (Number (Integer))
      created_at 1674596673 (Number (Integer))
      id "file-y4TRHmU2zS5tl45c0sRlFmnM" (Text)
      status_details null (Null)
      object "file" (Text)
      status "uploaded" (Text)
    Status Code:  200 (Number (Integer))
```

3. Use the “Create a job that fine-tunes…” (POST /fine-tunes) to create a fine-tuned model based on the uploaded file.

Important Note: Use the id received from the previous call as the “training_file” value



Connected System *

OpenAI CSP

Sort Endpoints Dropdown

fine-tune

Sort the endpoints dropdown below with a relevant search query.

Select Endpoint *

POST - Creates a job that fine-tunes a specified model from a given dataset. Response includes details...

POST /fine-tunes

Request Body

```
{
  training_file: "file-y4TRHmU2zS5tl45c0sRlFmnM",  /*Example: (Required) The
}
```

Success!

Time

391 ms

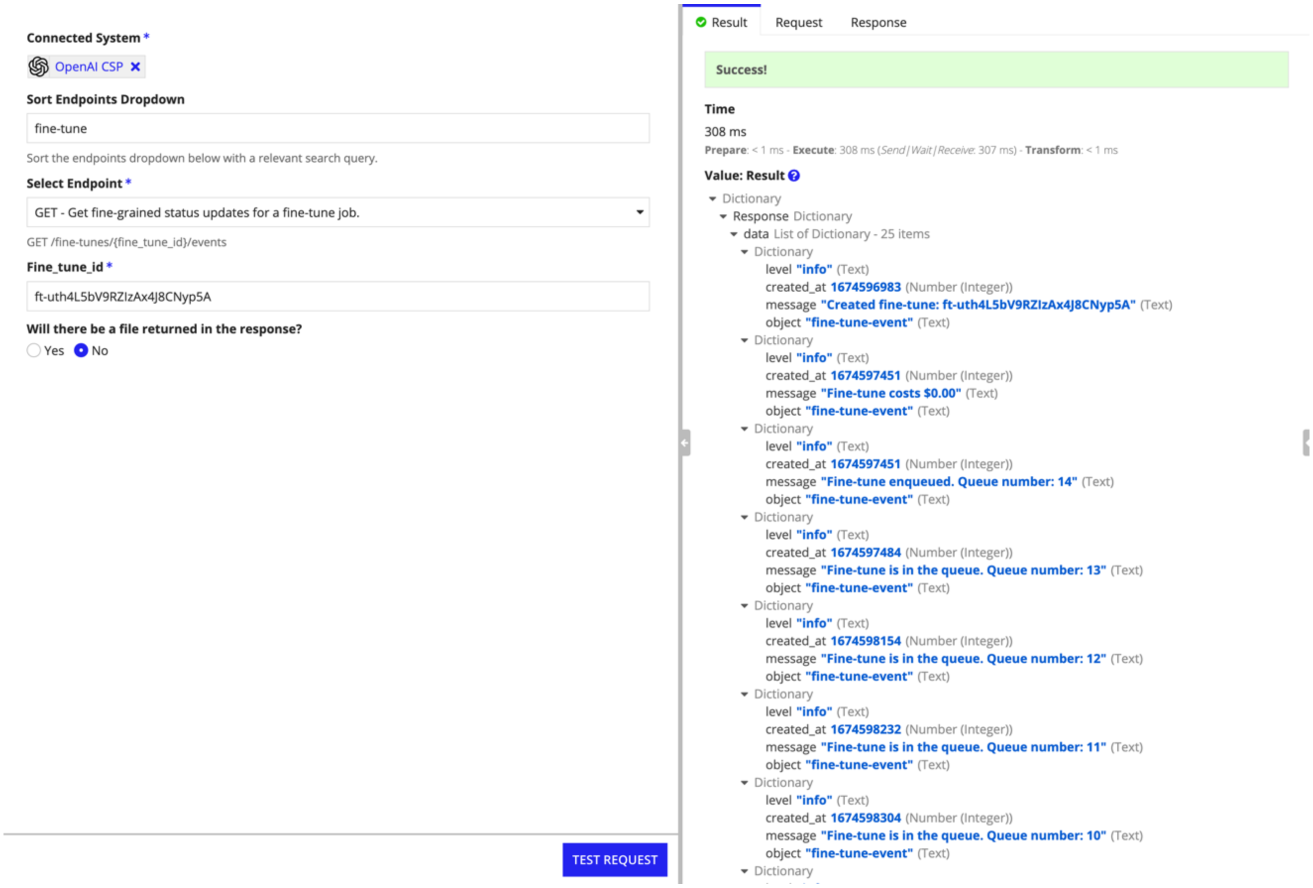
Prepare: < 1 ms - **Execute**: 391 ms (*Send / Wait / Receive*: 389 ms) - **Transform**: < 1 ms

Value: Result

```
▾ Dictionary
  ▾ Response Dictionary
      result_files List of Variant - 0 items
    ▾ hyperparams Dictionary
        n_epochs 4 (Number (Integer))
        batch_size null (Null)
        prompt_loss_weight 0.01 (Number (Decimal))
        learning_rate_multiplier null (Null)
      fine_tuned_model null (Null)
      created_at 1674596983 (Number (Integer))
    ▾ training_files List of Dictionary - 1 item
      ▾ Dictionary
          filename "sampleJSONFile7693419904657951 54.json" (Text)
          purpose "fine-tune" (Text)
          bytes 460 (Number (Integer))
          created_at 1674596673 (Number (Integer))
          id "file-y4TRHmU2zS5tl45c0sRlFmnM" (Text)
          status_details null (Null)
          object "file" (Text)
```

4. Check the status of the fine-tune job with the endpoint "Get info about the fine-tune…" (`GET /fine-tunes/{fine_tune_id}`). Status will be "pending" while the model is still training. The get status/info integrations will fail until status: "succeeded" If this is important for a workflow (say auto-creating a new model every month based on new data), you can create a process model to loop and continue checking the status of the model until it is ready to use. OpenAI does not currently have webhook functionality.

Important Note: After your job first completes, it may take several minutes for your model to become ready to handle requests. If completion requests to your model time out, it is likely because your model is still being loaded. If this happens, try again in a few minutes.



Connected System *

OpenAI CSP ✕

Sort Endpoints Dropdown

fine-tune

Sort the endpoints dropdown below with a relevant search query.

Select Endpoint *

GET - Get fine-grained status updates for a fine-tune job. ▾

GET /fine-tunes/{fine_tune_id}/events

Fine_tune_id *

ft-uth4L5bV9RZIzAx4J8CNyp5A

Will there be a file returned in the response?

○ Yes ● No

Success!

Time

308 ms

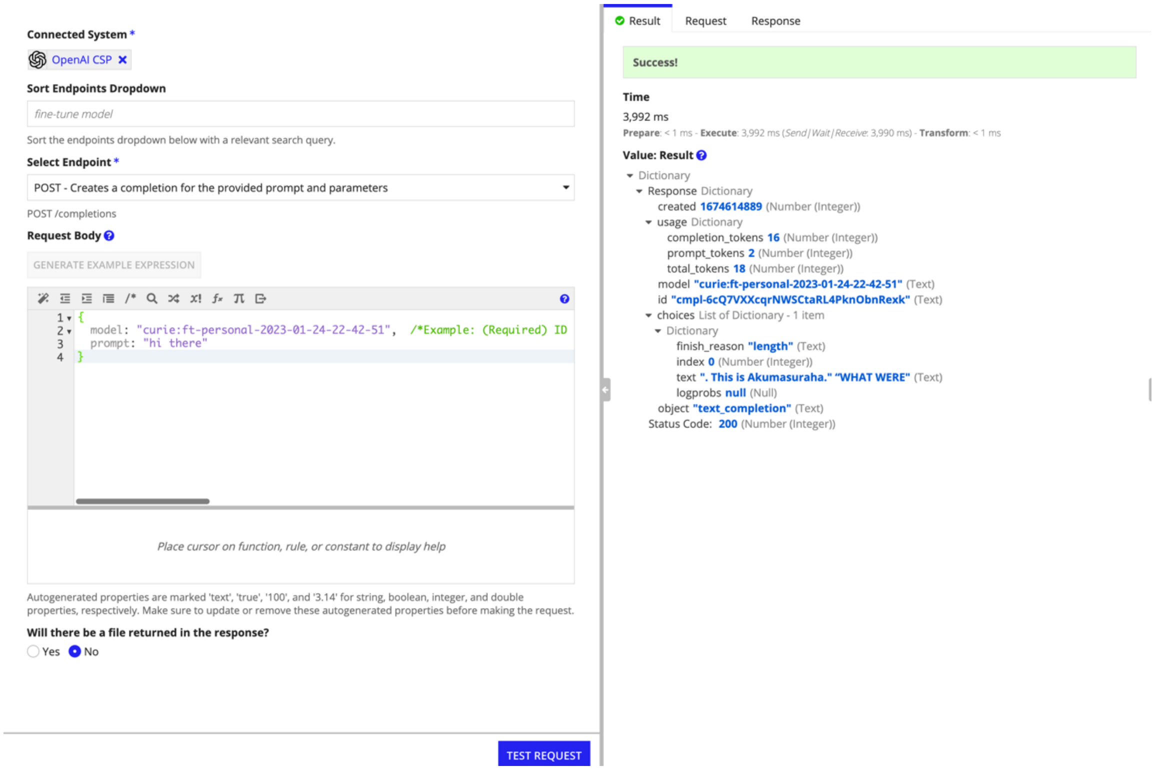
Prepare: < 1 ms - **Execute**: 308 ms (*Send / Wait / Receive*: 307 ms) - **Transform**: < 1 ms

Value: Result

```
▾ Dictionary
  ▾ Response Dictionary
    ▾ data List of Dictionary - 25 items
      ▾ Dictionary
          level "info" (Text)
          created_at 1674596983 (Number (Integer))
          message "Created fine-tune: ft-uth4L5bV9RZIzAx4J8CNyp5A" (Text)
          object "fine-tune-event" (Text)
      ▾ Dictionary
          level "info" (Text)
          created_at 1674597451 (Number (Integer))
          message "Fine-tune costs $0.00" (Text)
          object "fine-tune-event" (Text)
      ▾ Dictionary
          level "info" (Text)
          created_at 1674597451 (Number (Integer))
          message "Fine-tune enqueued. Queue number: 14" (Text)
          object "fine-tune-event" (Text)
      ▾ Dictionary
          level "info" (Text)
          created_at 1674597484 (Number (Integer))
          message "Fine-tune is in the queue. Queue number: 13" (Text)
          object "fine-tune-event" (Text)
      ▾ Dictionary
          level "info" (Text)
          created_at 1674598154 (Number (Integer))
          message "Fine-tune is in the queue. Queue number: 12" (Text)
          object "fine-tune-event" (Text)
      ▾ Dictionary
          level "info" (Text)
          created_at 1674598232 (Number (Integer))
          message "Fine-tune is in the queue. Queue number: 11" (Text)
          object "fine-tune-event" (Text)
      ▾ Dictionary
          level "info" (Text)
          created_at 1674598304 (Number (Integer))
          message "Fine-tune is in the queue. Queue number: 10" (Text)
          object "fine-tune-event" (Text)
      ▾ Dictionary
```

5. When a job has succeeded, the fine_tuned_model field will be populated with the name of the model. You may now specify this model as a parameter to the Completions API (POST /completions, …).

After your job first completes, it may take several minutes for your model to become ready to handle requests. If completion requests to your model time out, it is likely because your model is still being loaded. If this happens, try again in a few minutes. You can start making requests by passing the model name as the model parameter of a completion request:



Connected System *

OpenAI CSP ✕

Sort Endpoints Dropdown

fine-tune model

Sort the endpoints dropdown below with a relevant search query.

Select Endpoint *

POST - Creates a completion for the provided prompt and parameters

POST /completions

Request Body

```
{
  model: "curie:ft-personal-2023-01-24-22-42-51",  /*Example: (Required) ID
  prompt: "hi there"
}
```

Place cursor on function, rule, or constant to display help

Autogenerated properties are marked 'text', 'true', '100', and '3.14' for string, boolean, integer, and double properties, respectively. Make sure to update or remove these autogenerated properties before making the request.

Will there be a file returned in the response?

○ Yes ● No

Success!

Time

3,992 ms

Prepare: < 1 ms - **Execute**: 3,992 ms (*Send/Wait/Receive*: 3,990 ms) - **Transform**: < 1 ms

Value: Result

```
Dictionary
  Response Dictionary
    created 1674614889 (Number (Integer))
    usage Dictionary
      completion_tokens 16 (Number (Integer))
      prompt_tokens 2 (Number (Integer))
      total_tokens 18 (Number (Integer))
    model "curie:ft-personal-2023-01-24-22-42-51" (Text)
    id "cmpl-6cQ7VXXcqrNWSCtaRL4PknObnRexk" (Text)
    choices List of Dictionary - 1 item
      Dictionary
        finish_reason "length" (Text)
        index 0 (Number (Integer))
        text ". This is Akumasuraha." “WHAT WERE" (Text)
        logprobs null (Null)
    object "text_completion" (Text)
  Status Code:  200 (Number (Integer))
```