Document Vector Database Connected System for Appian

V3.2.6

Appian Corporation

Version 3.2.3

Table of Contents

Overview

The Document Vector Database Connected System enables Large Language Models (LLMs) to answer user submitted questions based on Appian Knowledge Base Documents. By uploading documents to this connected system, users can perform semantic searches to pinpoint the most pertinent content related to their questions. The Connected System also boasts Client APIs tailored for the AI Knowledge Assistant Component. This allows the AI Knowledge Assistant Component to deliver AI generated answers to user inquiries sourced from documents stored in the database, as well as general questions.

Note: This plugin may not function as expected in High Availability(HA) environments.

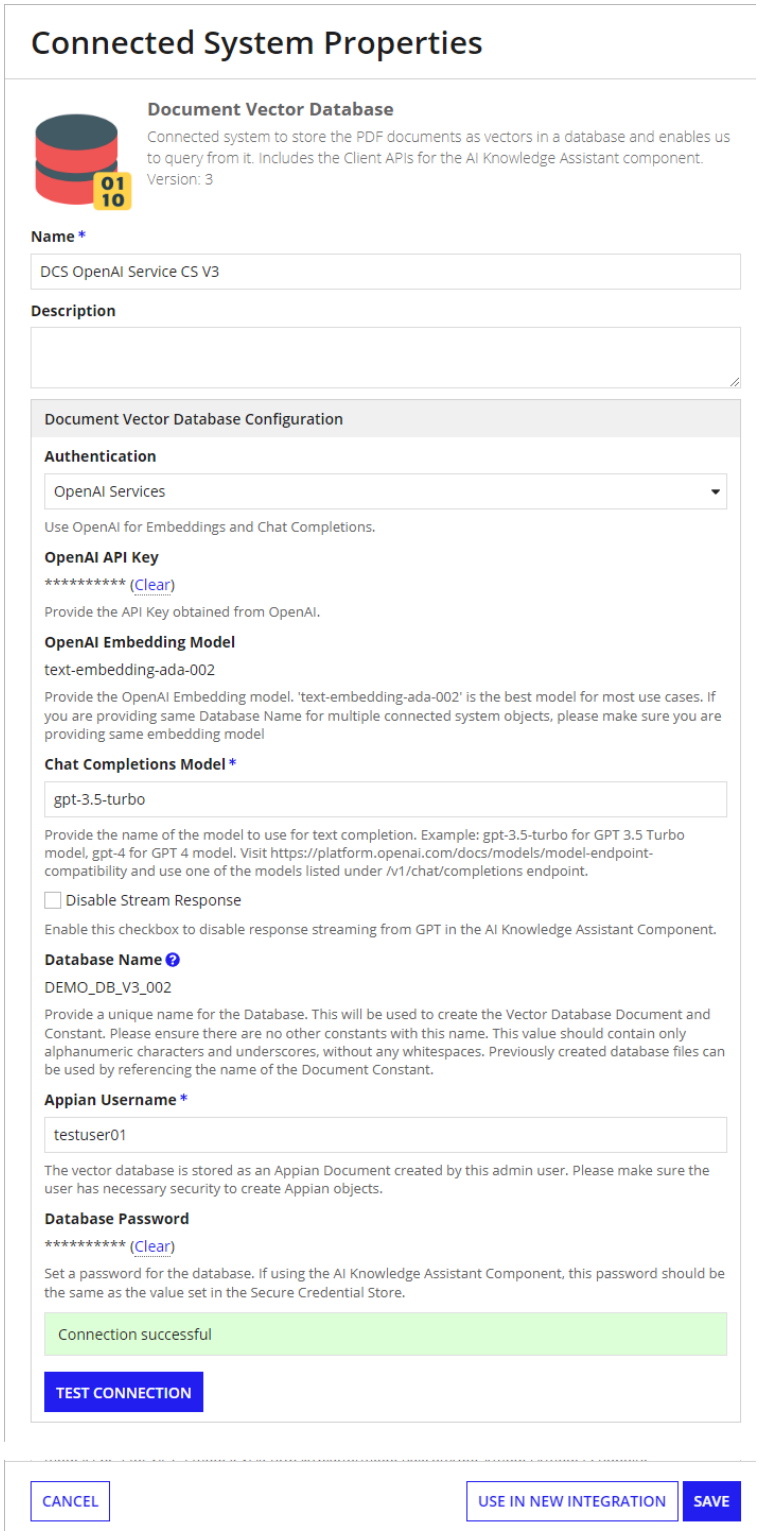
If you haven’t worked with vector databases or large language models before, here is a brief metaphor that should help explain the interaction:

- Imagine a vast library filled with thousands of books. Each book in this library represents an Appian Knowledge Base document. Now, instead of meticulously scanning each book from cover to cover every time you seek an answer, consider a different approach: for every book, we take multiple snapshots of its pages. These aren't visual images; instead, they capture the essence or meaning of each page's content. These are called "text embeddings" in our vector database.
- Each page can be broken down (chunked) into several of these embeddings, ensuring that even the subtle nuances or topics within a page aren't missed.
- When a query arises, there's no need to go through every book. Simply reviewing these embeddings can help identify which pages or sections might hold the answer. After narrowing down to these potential pages, a highly knowledgeable librarian (our large language model) instantly processes the content and offers a relevant response, directly sourced from our documents.
- In essence, the vector database efficiently navigates the vast sea of information in our library, and the language model extracts and delivers the exact knowledge you're after.

Features

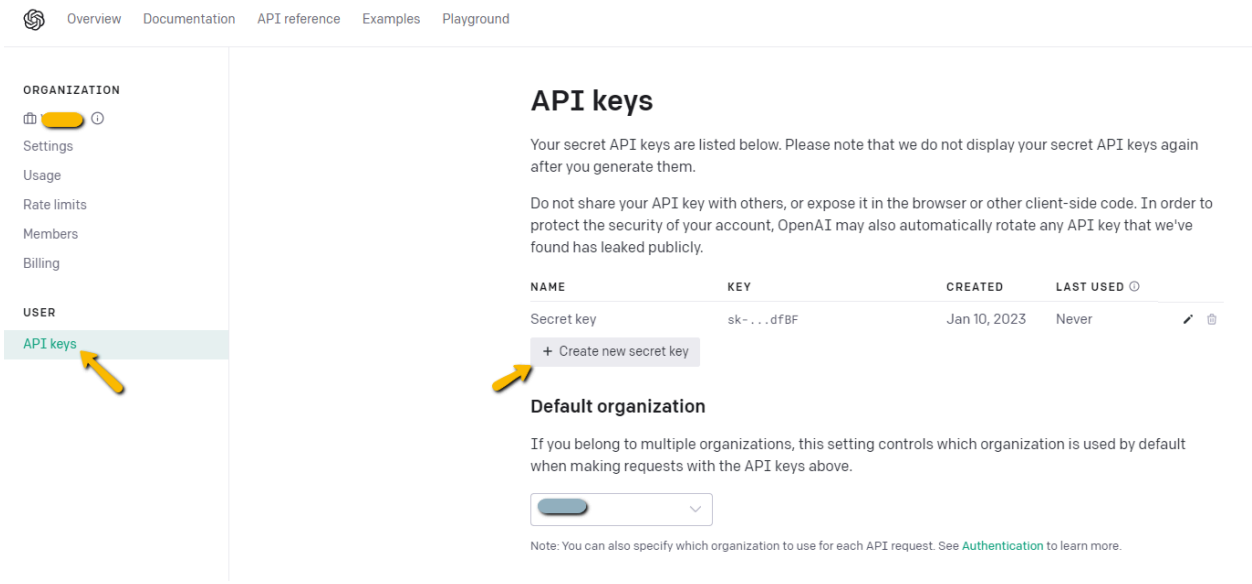
- Upload Document - Uploads and stores the documents and its vectors in the database.
- List Documents - Provides a list of documents uploaded in the database.
- Database Operations
    - Delete Documents - Deletes a given document that has been uploaded in the database.
    - Sync Documents - Updates the existing documents in the database with the latest version of the document available in the Appian Knowledge Center.
    - Change Database Password - Changes Database password.
- Query Documents - Get relevant pieces of content from documents for the given prompt.
- Generate Response - Perform search in the given documents and generate a ChatGPT response for the given prompt.
- Group Text - Groups the given List of Text into the provided List of Categories based on relevancy.
- Servlets for AI Knowledge Assistant - fetches the logged in username(encrypted), performing completions(chat and document querying) and uploading new documents to the database.
- Client APIs for AI Knowledge Assistant component - fetches document details, fetches connected system credentials(encrypted), performing completions(chat and document querying) and uploading new documents to the database.

Using OpenAI Services



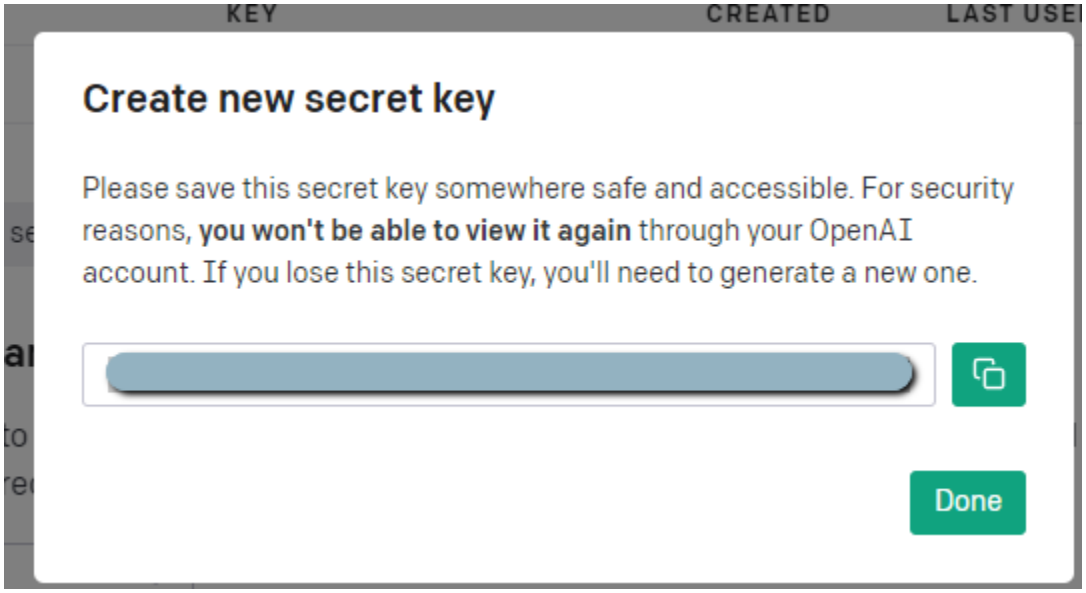
The connected system using the OpenAI configuration requires the following values: OpenAI API Key, OpenAI Embedding Model name,OpenAI Chat Completion Model name, Disable Stream Response, Database Constant Name, Database Password, and Appian Username.

1. Go to the OpenAI console. Make sure that the **API keys** menu is selected.



Click on **Create new secret key** to generate a new API key.

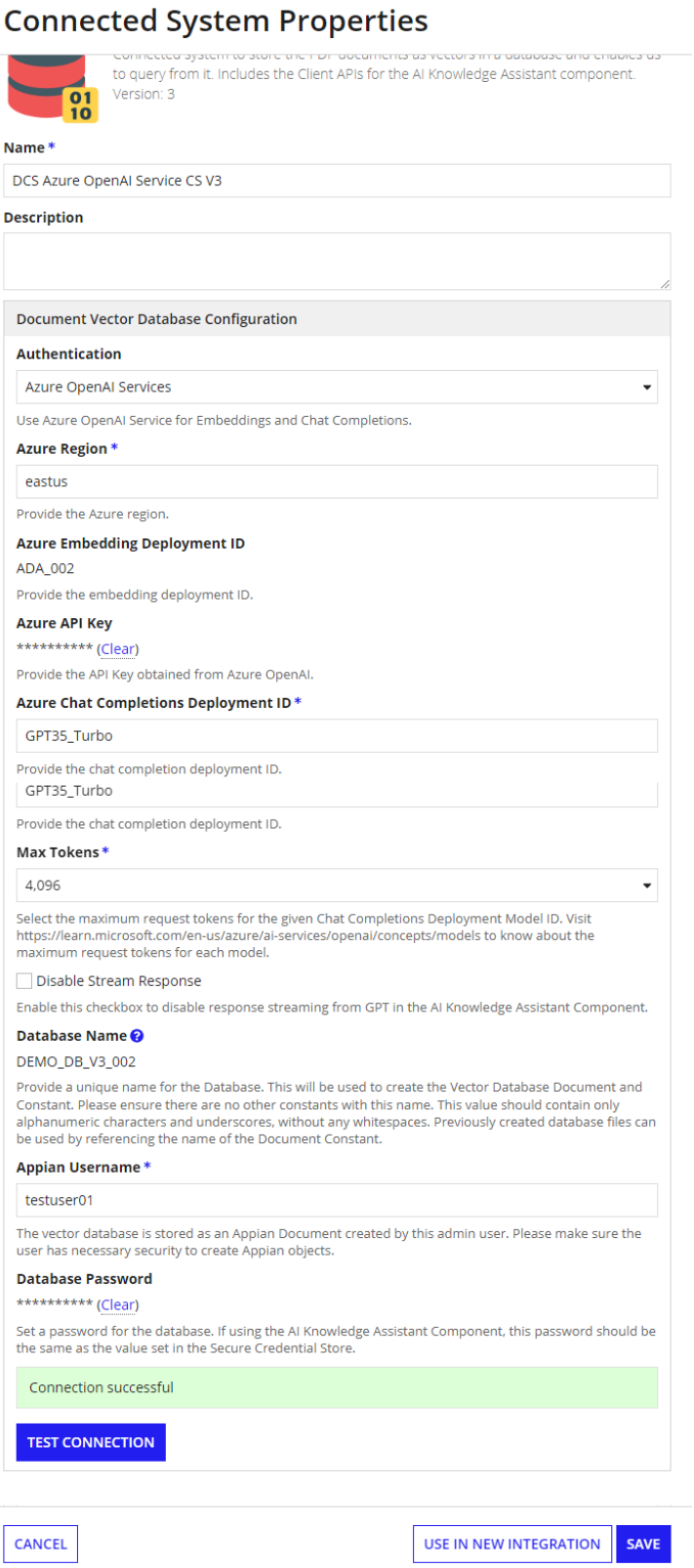
1. Copy the value and save it separately as we won't be able to access it again. Paste the API key in the connected system dialog box.



2. Refer embeddings models for a model name.

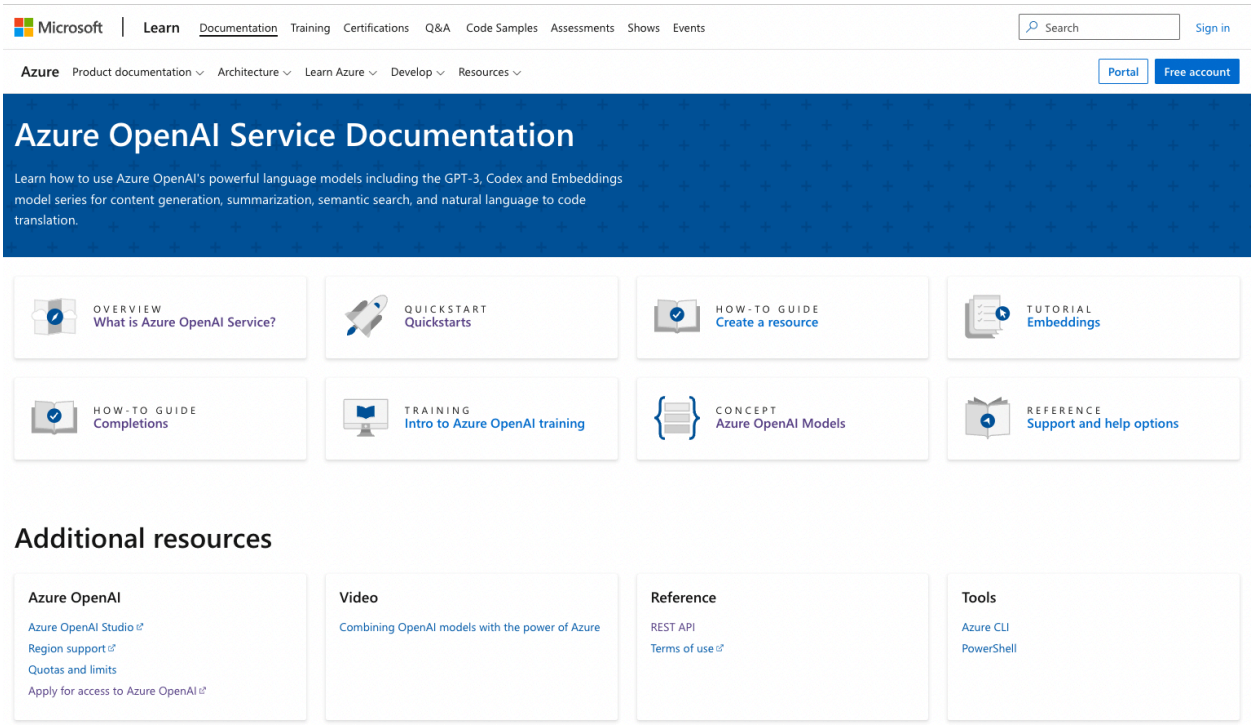
3. Provide the Database name which will be used in the creation of the Database file in Appian. This is stored as an Appian Document.
4. Provide the OpenAI completions model which will be used for the Generate Response integration, Servlet and Client API. It is recommended to use "gpt-4" based models to get more relevant and accurate responses.
5. If you are using the AI Knowledge Assistant component and need to disable the streaming responses(which uses servlets), select the "Disable Stream Response" in the configuration. This will enable the component to use Client APIs and doesn't provide streaming responses. By default, Stream Response will be enabled.
6. Provide the Appian Username which will be used to create the database document and the associated Appian Objects. This user must be a System Administrator user.
   For best practices, create a Service Account and update the user type to System Administrator and provide the username of the Service Account here.
7. Provide the Database Password which will be used in the configuration of the Database file. Make sure to remember this value—this password is required to set up document security in the AI Knowledge Assistant component.
8. Once you have entered all the credentials, click on **Test Connection.** Success message will be shown if the provided credentials are correct.

Using Azure OpenAI Services

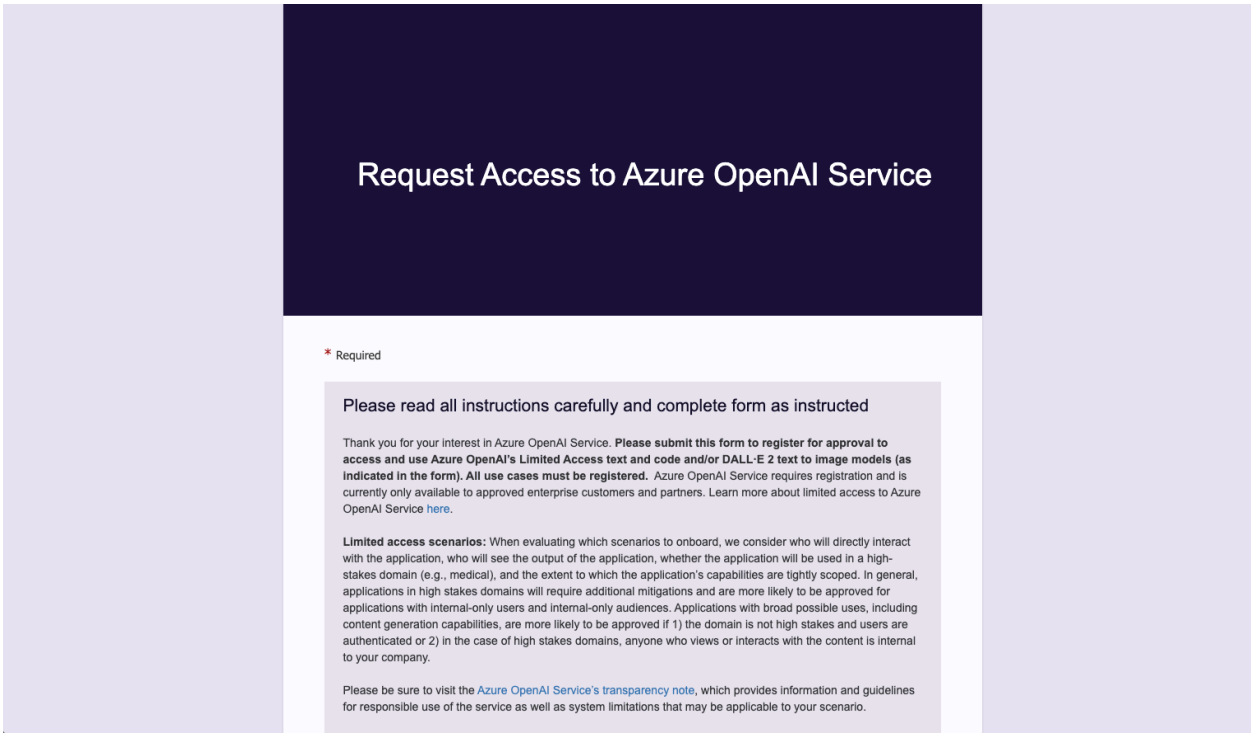


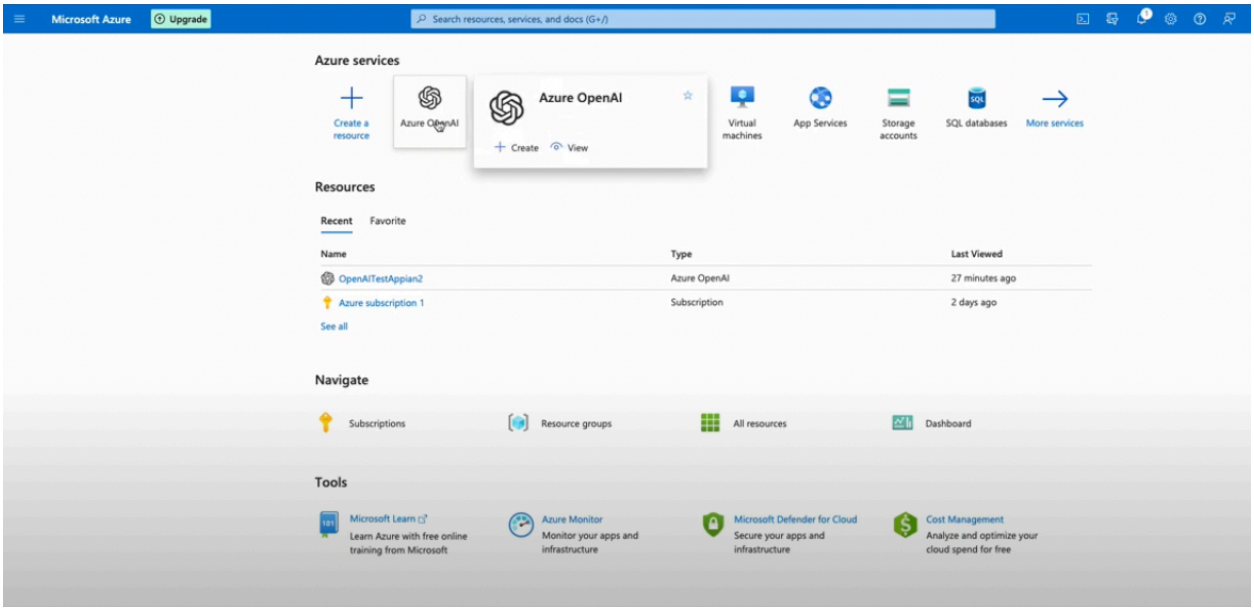
The connected system using the Azure OpenAI configuration requires the following values: Azure Region, Azure Deployment ID, Azure API Key, Azure Chat Completions Deployment ID, Max Tokens, Disable Stream Response, Database Name, Appian Username, and Database Password.

1. Navigate to Azure's OpenAI API docs and ensure you have met the listed prerequisites. View the prerequisites by selecting "Quickstarts." If you have not already done so, create an Azure subscription.

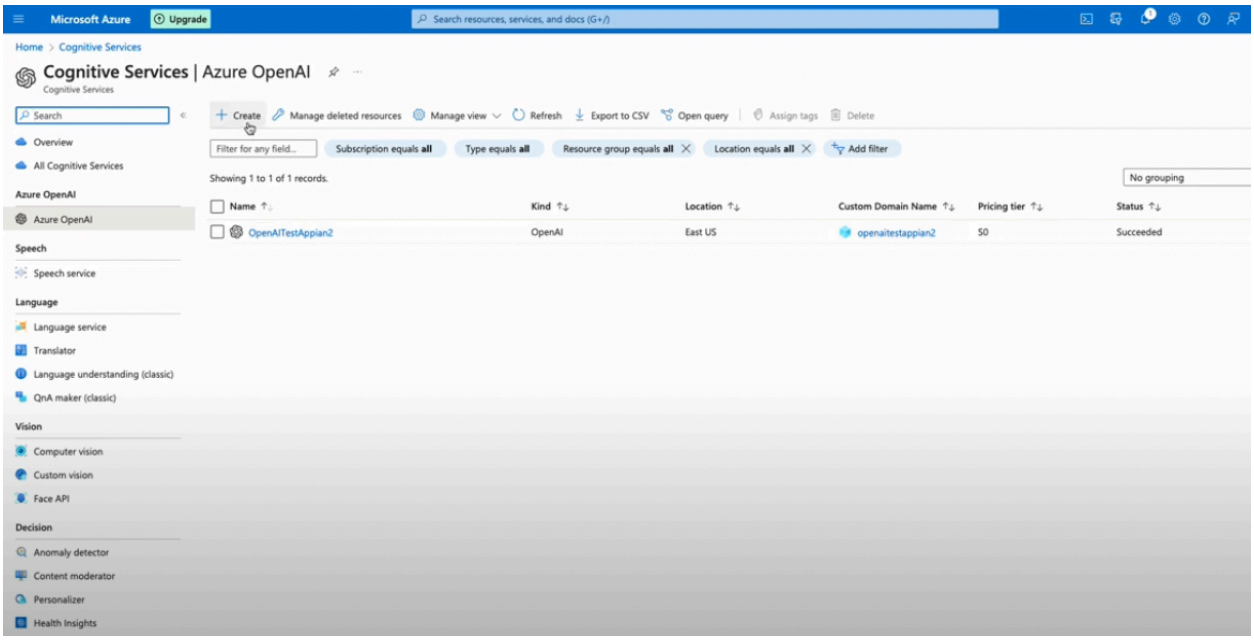


2. Apply for access to Azure OpenAI services by completing the form here. You will need your subscription ID from the previous step

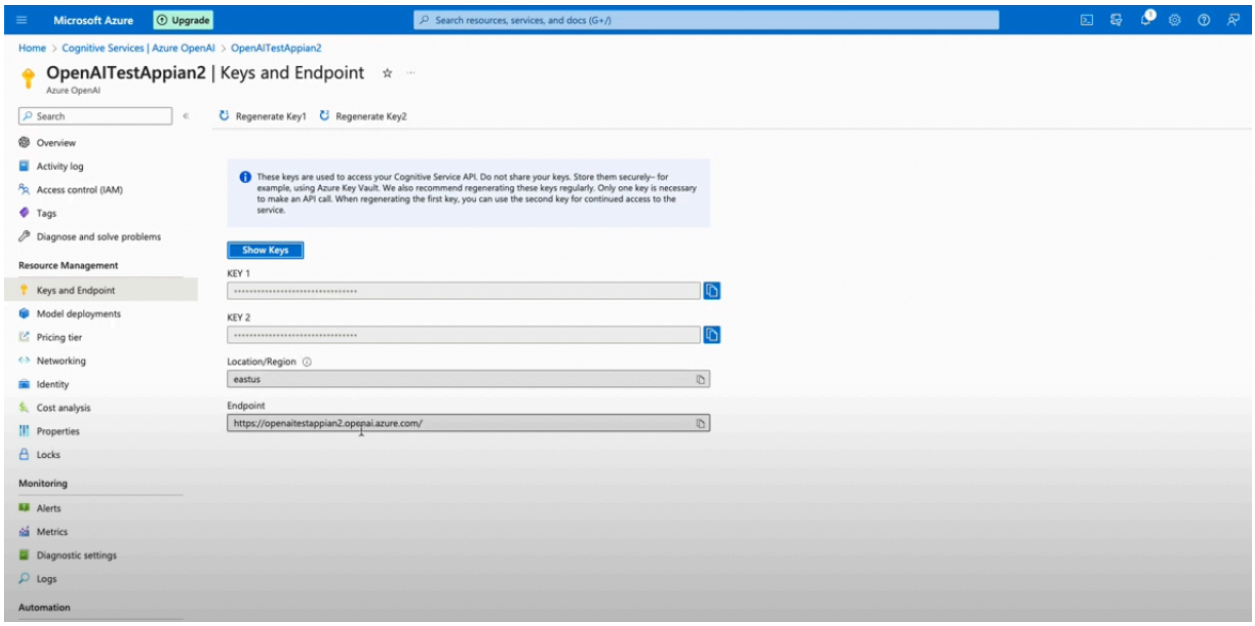




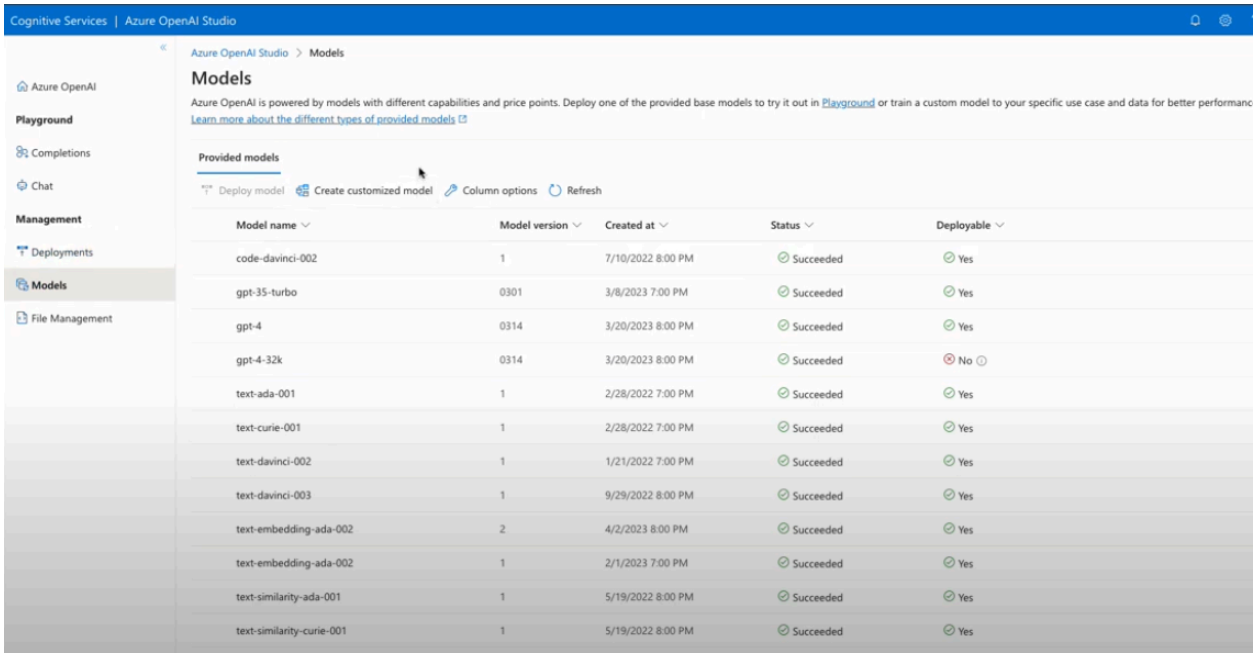
3. Create a service and set a domain name.



4. Within your service, create and access API keys through "Keys and Endpoints" under Resource Management.



5. Deploy OpenAI models through the Azure OpenAI Studio. Information generated from these deployments will be needed to use the connected system. This information can be accessed through the "Deployments" tab in Azure OpenAI Studio.



6. Follow the steps given in the previous type of connected system to configure the Disable Stream Response, Database Name, Appian Username, and Database Password.

7. Once you have entered all the credentials, click on **Test Connection.** Success message will be shown if the provided credentials are correct.

Production Usage

IMPORTANT NOTE: If you are using this plugin in production, open a support case and ask to increase Heap Max for app server by 1GB. This will increase query performance and allow the plugin to handle a larger number of concurrent users.

Migration Guide

If you are already using an older version of the Document Vector Database Connected System, complete the following steps to migrate to the newer version. This will be a one-time setup.

- Once the latest version of the connected system is installed, create a new Vector Database Connected System object. The previous connected system object and integrations based on it will continue to work.
  Important Note: While creating the connected system object, do not use the same Database Name, provide a different value.
- Use the List Documents Integration (based on the older version of the connected system) to get the list of all documents uploaded to the previous vector database.
- Create an integration based on the new connected system object using the Upload Document Integration template and upload all the documents that were in the previous database.
- Update the integration objects used in your application with the newly created connected system. Click the "x" next to the connected system name in the integration to swap out the old connected system object to the new one.
- If you are using the "AI Knowledge Assistant Component", update the component to the latest version. This is required for it to work with the new version of the connected system. After updating the component to the latest version, follow the Migration Guide provided in the component documentation.

IMPORTANT NOTE: When migrating/deploying the application from one environment to another, all documents that have been previously uploaded to the vector database in the older environment will not be present in the new vector database. Since documents do not retain their document ID across environments, you will need to upload each document again to the new vector database.

Integration Configuration

Note:
"Logged In User" input field must be set with the value "loggedInUser()" to properly set up document security. This will enable users to upload, delete, and view documents they have access to in the Appian Knowledge Center.

Upload Document

This integration vectorizes and stores the document in a native Appian vector database. After uploading the document, use the "Query Documents" or "Generate Response" integration to semantically search the document.

Note: If a new version of a document is uploaded to Appian, make sure to use this integration to re-index the vector database with the updated version. This will overwrite the existing document information and ensure that the vector database is sourcing information from the most up-to-date versions.

- If a user overwrites an existing Appian Knowledge Base document with a new version (i.e. Appian document is overwritten with an updated document and the document ID does not change), then simply run this integration with the new version of the document.
- If a user creates a new document (i.e. a new file is created with a new document id), then use the "Update Previously Embedded Document" parameter to select which preexisting document should be replaced with the new version of the document.
- Use the "Sync Documents" integration to bulk upload the newest versions of all documents to the vector database.

Inputs:

Input Document(Document) - Required

Description: Provide the document to upload. Supported file type: PDF.

Update Previously Embedded Document(Boolean) - Optional

Description: Provide true to replace an already uploaded document.

Document To Be Replaced(Document) - Required when *Update Previously Embedded Document* is true.

Description: Provide the document to be replaced. This document will be deleted from the database.

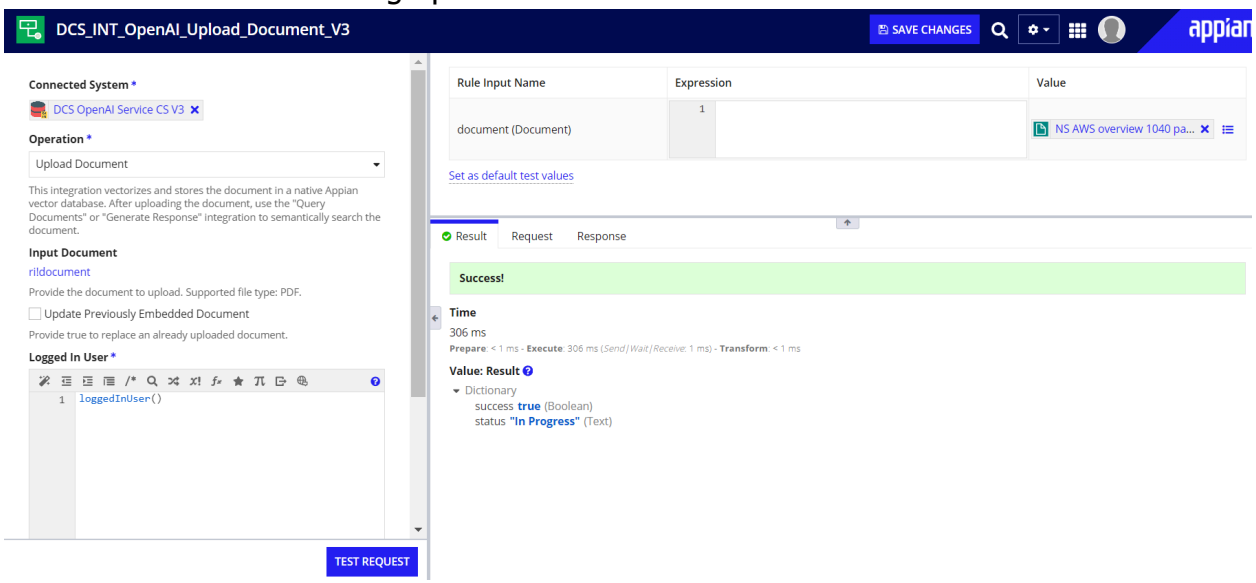
Logged In User(Text) - Required.

Description: Provide the logged in user with the loggedInUser() function from the expression box. This is required to set up user-level document security.

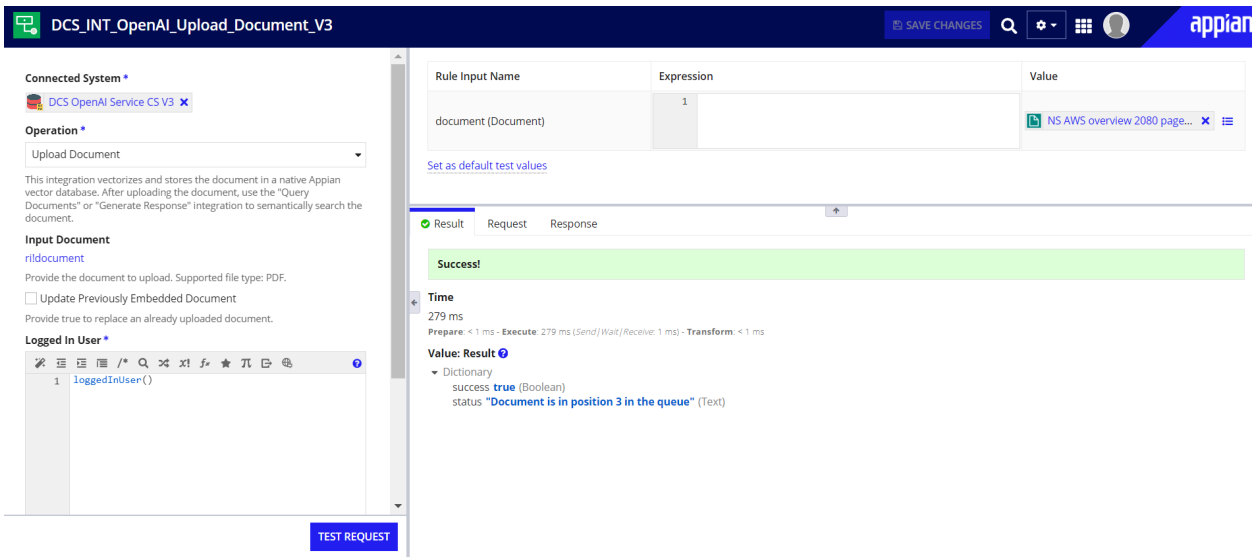
Note: This Integration uploads only a single document at a time. The other documents will be added to the queue. Run List Documents Integration to get the upload status of the documents.

Output:

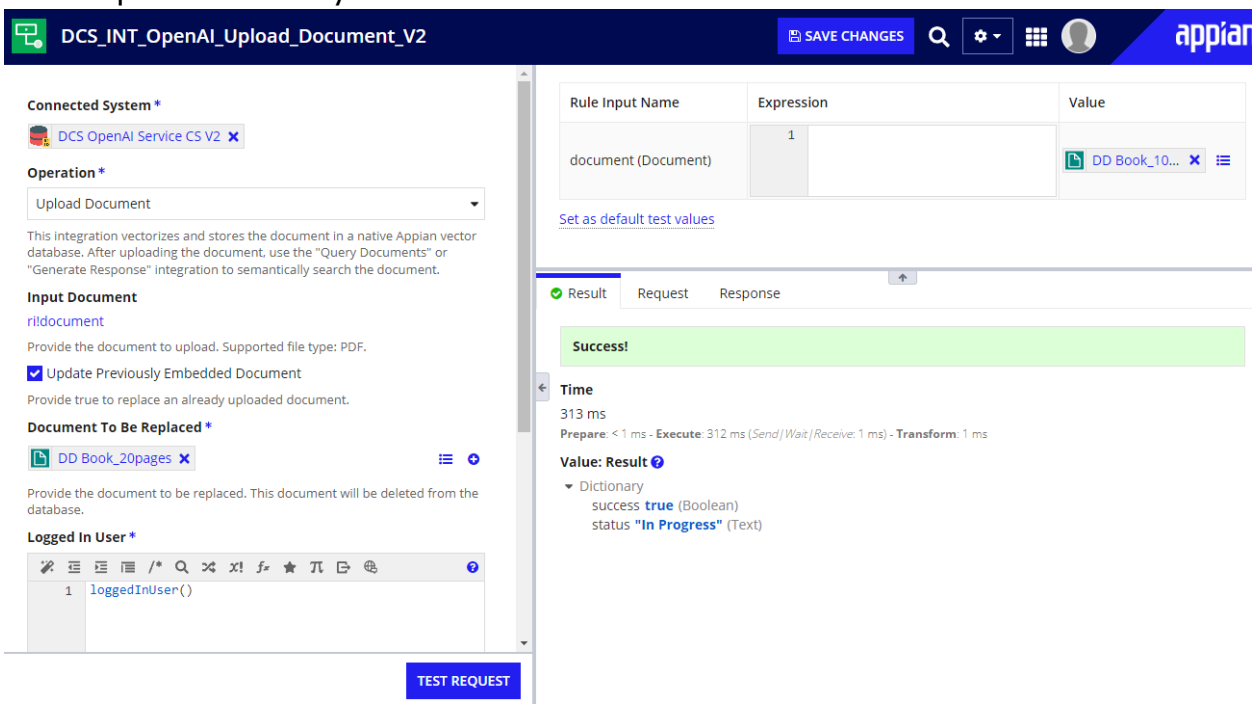
1. No document is being uploaded



2. Some documents are present in the queue



3. Update Previously Embedded document



List Documents

This integration lists the names and IDs of the documents stored in the vector database.

Inputs:

Limit (Number(Integer)) - Optional

Description: Provide the number of documents to return from the database.Default: 1000.

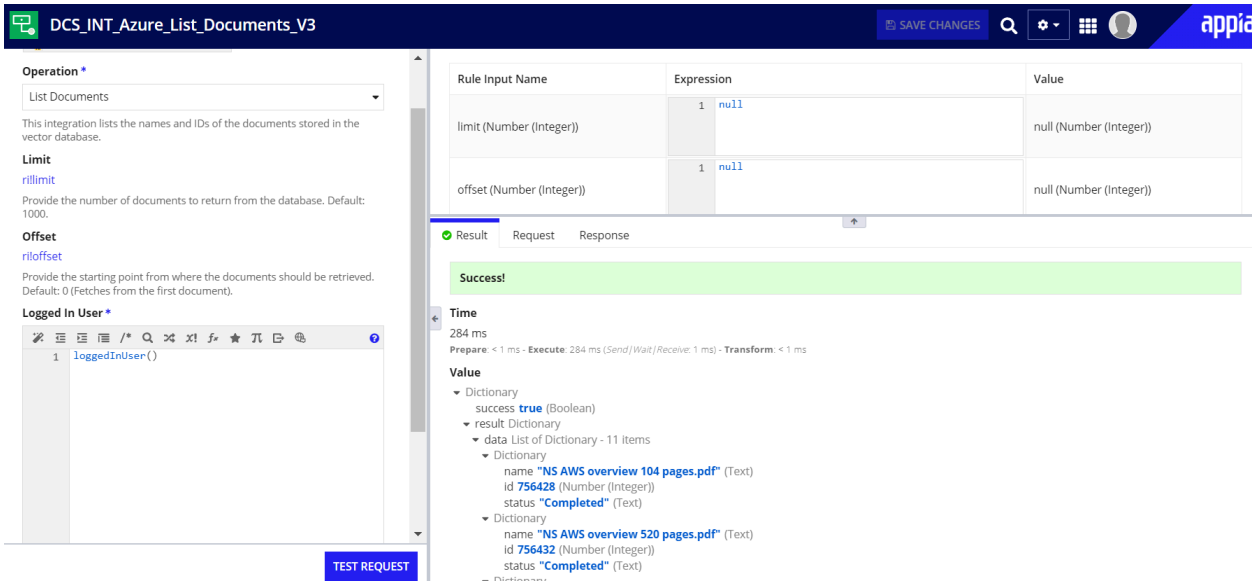
Offset (Number(Integer)) - Optional

Description: Provide the starting point from where the documents should be retrieved. Default: 0 (Fetches from the first document).

Logged In User(Text) - Required.

Description: Provide the logged in user with the loggedInUser() function from the expression box. This is required to set up user-level document security.

Output:



Database Operations

This integration performs the following database operations on the vector database: Delete Documents, and Change Database Password.

- **Delete Documents**

Deletes the documents uploaded to the database based on the conditions and list of document IDs.

Inputs:

Condition (Text) - Required

Description: Provide the condition for deleting the list of documents from the Database. Valid values: IN, NOT IN, ALL.

Notes:

IN - Delete the documents that are given in the list of document IDs input.
NOT IN - Delete the documents that are NOT in the given list of document IDs input.
ALL - Delete all the documents in the database.

Documents (List of Documents) - Required(when the condition is IN or NOT IN)

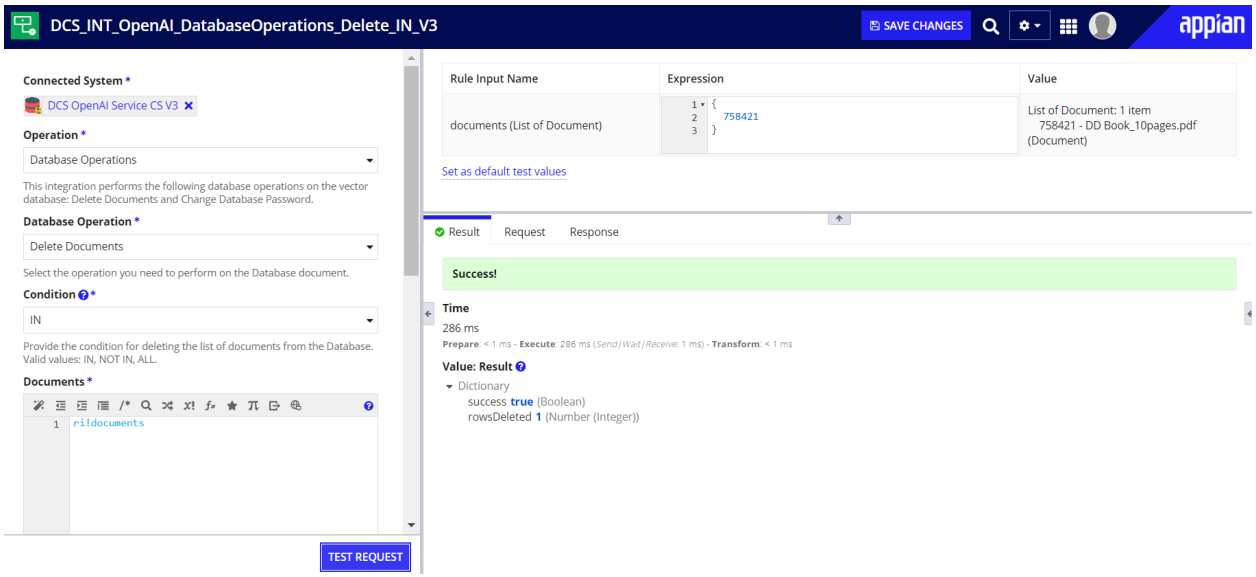
Description: Provide the List of Document IDs to perform the delete operation.

Logged In User(Text) - Required.

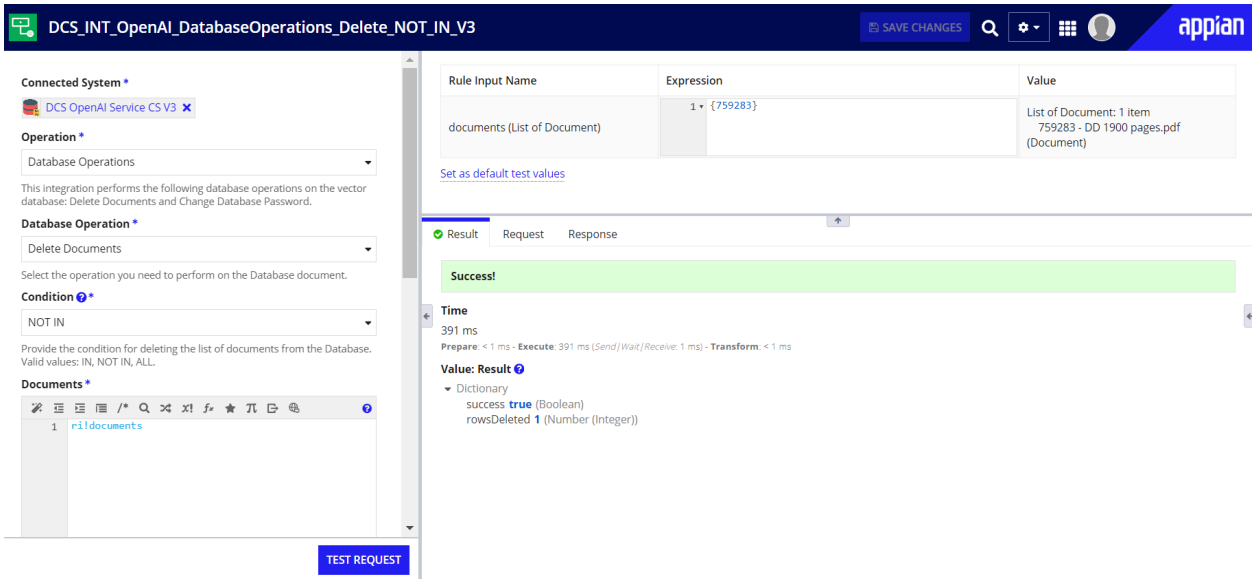
Description: Provide the logged in user with the loggedInUser() function from the expression box. This is required to set up user-level document security.

Output:

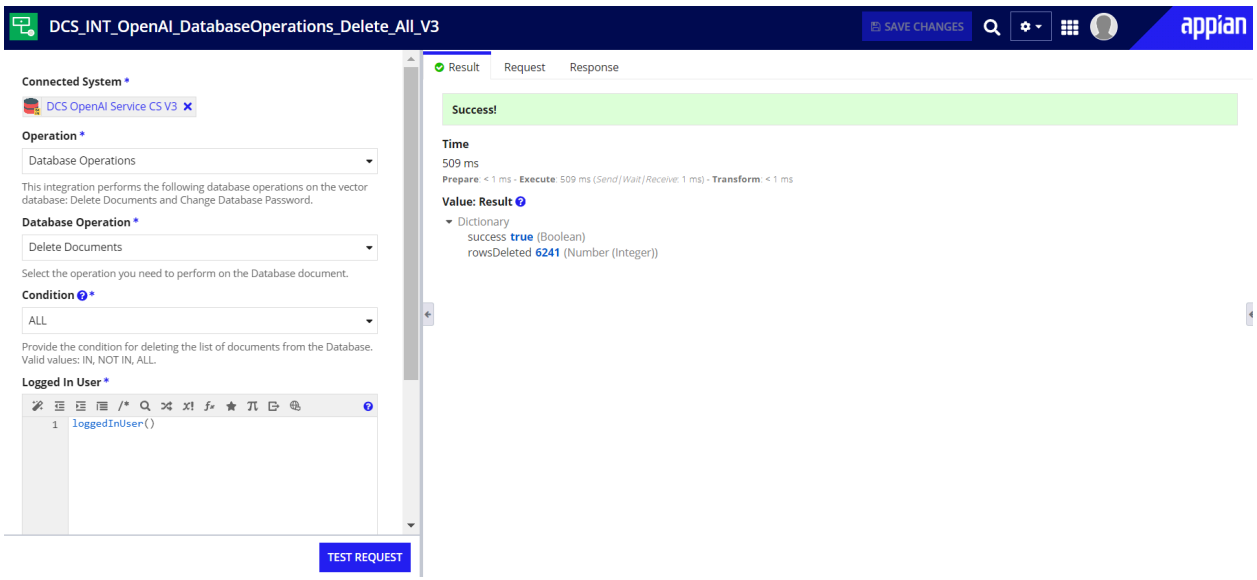
IN condition:



NOT IN condition:



ALL condition:



- **Sync Documents**

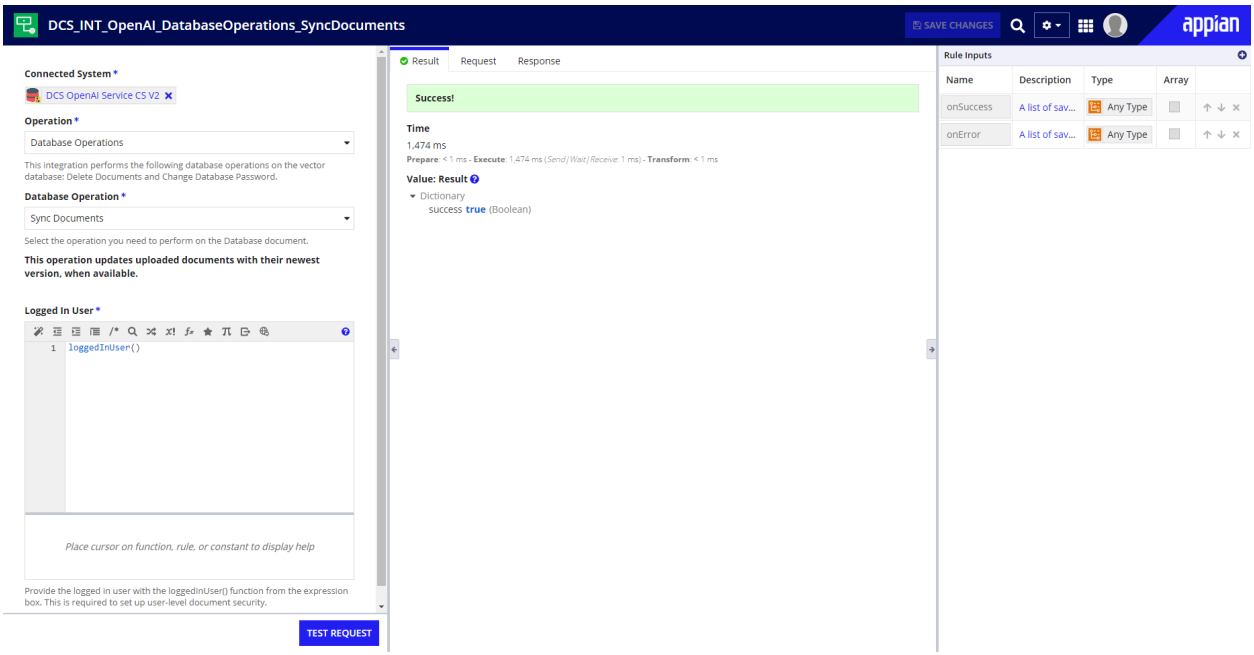
This integration updates uploaded documents with their newest version, when available.

Inputs:

Logged In User(Text) - Required.

Description: Provide the logged in user with the loggedInUser() function from the expression box. This is required to set up user-level document security.

Output:



- **Change Password**

To change the password for the database. Once changing you need to update the connected system configuration to use the new password.

Inputs:

Old Password(Text) - Required

Description: Provide the existing database password.

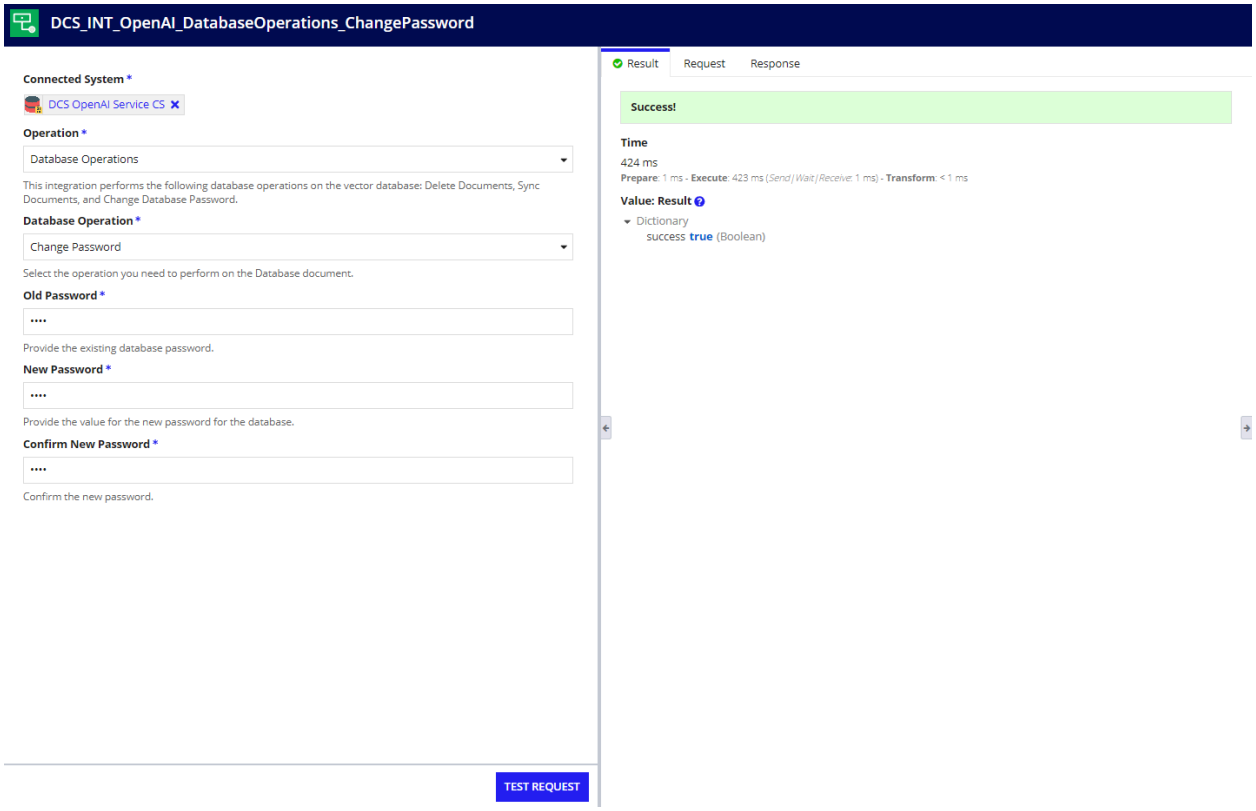
New Password(Text) - Required

Description: Provide the value for the new password for the database.

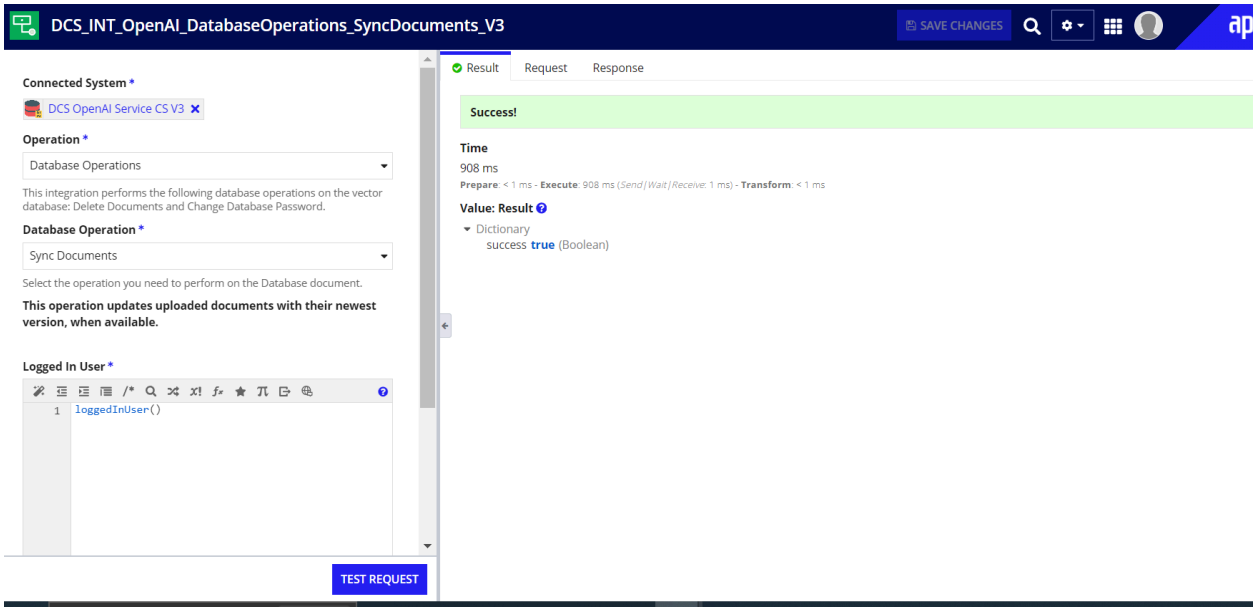
Confirm New Password(Text) - Required

Description: Re-enter the new password to confirm.

Output:



Output:



Query Documents

This integration semantically searches the documents in the vector database and finds the most relevant matches to the query.

Inputs:

Query (Text) - Required

Description: Provide a query for semantically searching the vector database.

Documents (List of Documents) - Required

Description: Provide the list of Documents or Document IDs to query from within the Vector Database. If no document is specified, it will query from the entire database.

Include Embeddings(Boolean) - Optional

Description: Provide true to get the text embeddings in the output.Default: false

Top K(Number(Integer)) - Optional

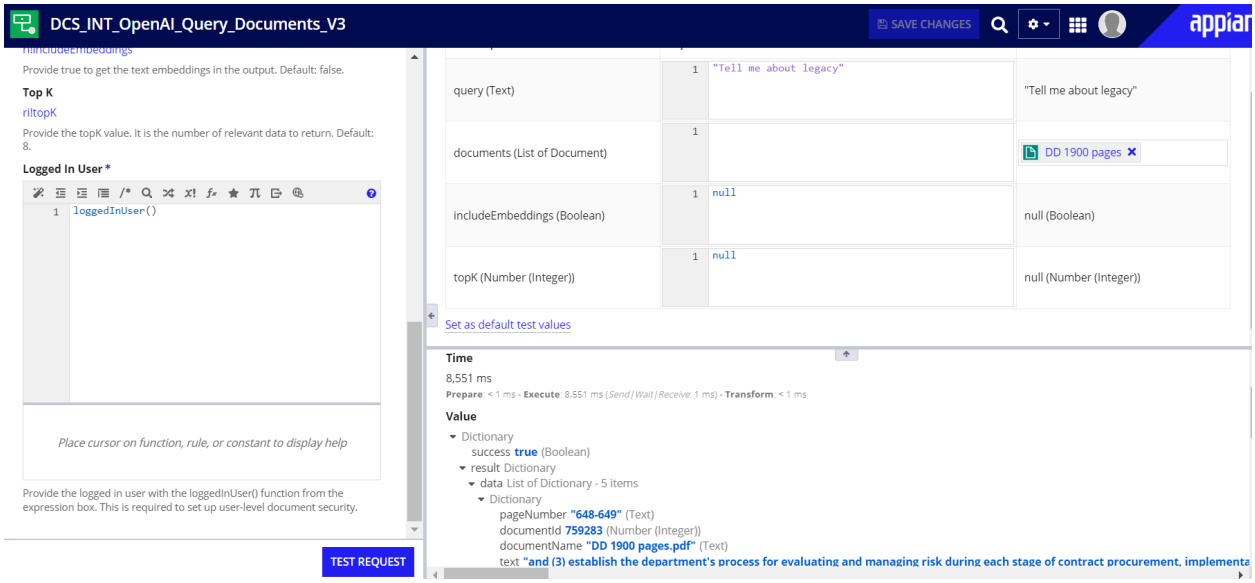
Description: Provide the number of relevant data to return. Refer Top K Configuration for configuring Top K

Logged In User(Text) - Required.

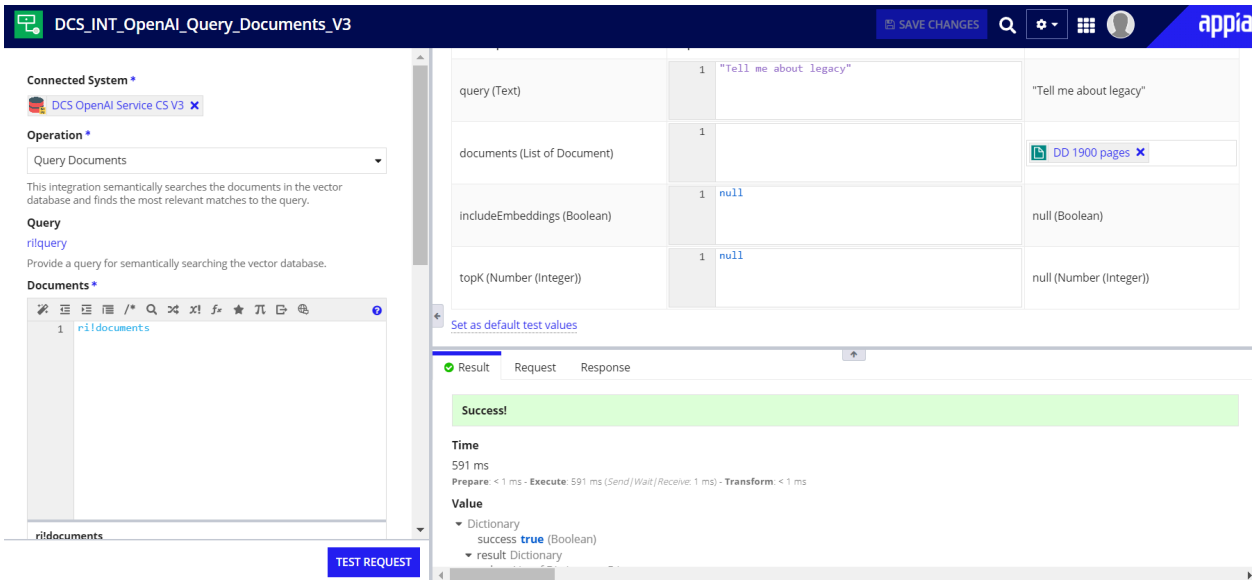
Description: Provide the logged in user with the loggedInUser() function from the expression box. This is required to set up user-level document security.

Output:

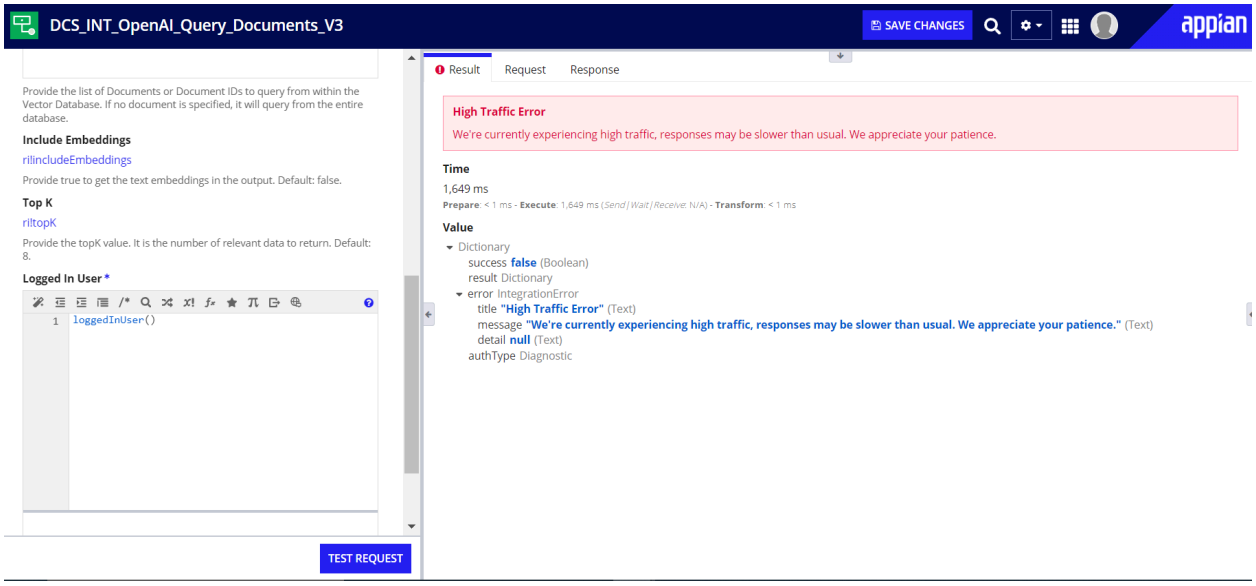
1. Querying when the document is not present in cache



2. Querying when the document is present in cache



3. Error when sufficient memory is not present in cache.



Generate Response

This integration semantically searches the documents in the vector database and finds the most relevant matches to the query. It then prepends this document information to the user's query and sends it to an OpenAI completion model. OpenAI will provide the answer to the query using the relevant document information.

Inputs:

Query (Text) - Required

Description: Provide the query.

Documents (List of Documents) - Required

Description: Provide the list of Documents or Document IDs to query from within the Vector Database. If no document is specified, it will query from the entire database.

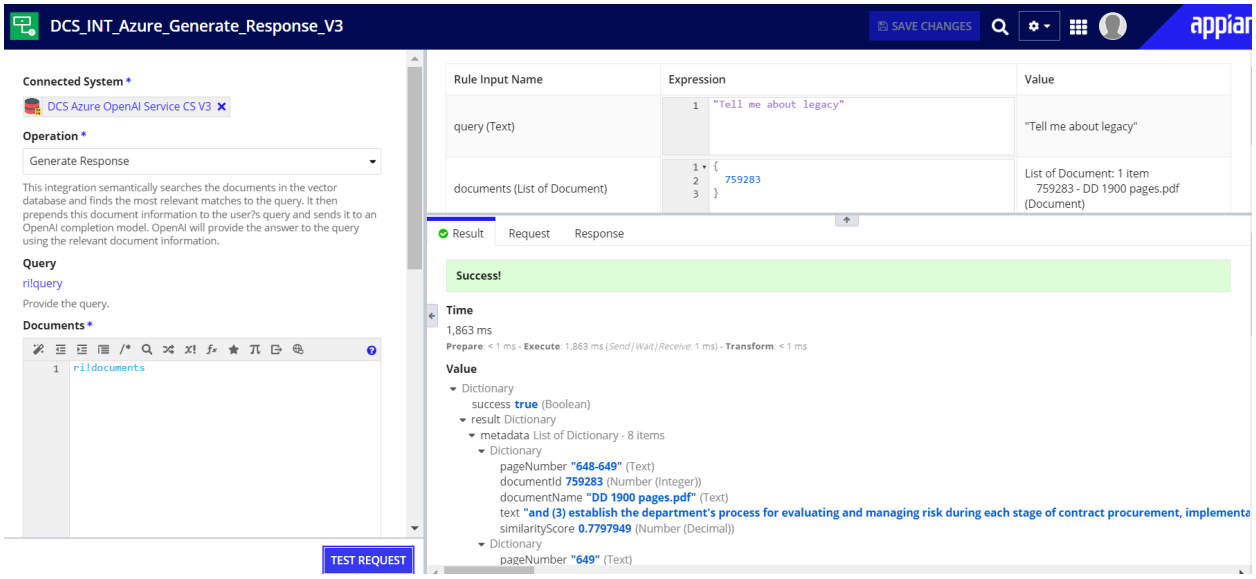
Top K(Number(Integer)) - Optional

Description: Provide the number of relevant data to return. Refer Top K Configuration for configuring Top K

Logged In User(Text) - Required.

Description: Provide the logged in user with the loggedInUser() function from the expression box. This is required to set up user-level document security.

Output:



Group Text

This integration groups the given List of Text into the provided List of Categories based on relevancy.

Inputs:

Text to Group(List of Text) - Required

Description: Provide the list of Text to be categorized.

Categories to Group (List of Text) - Required

Description: Provide the list of Categories.

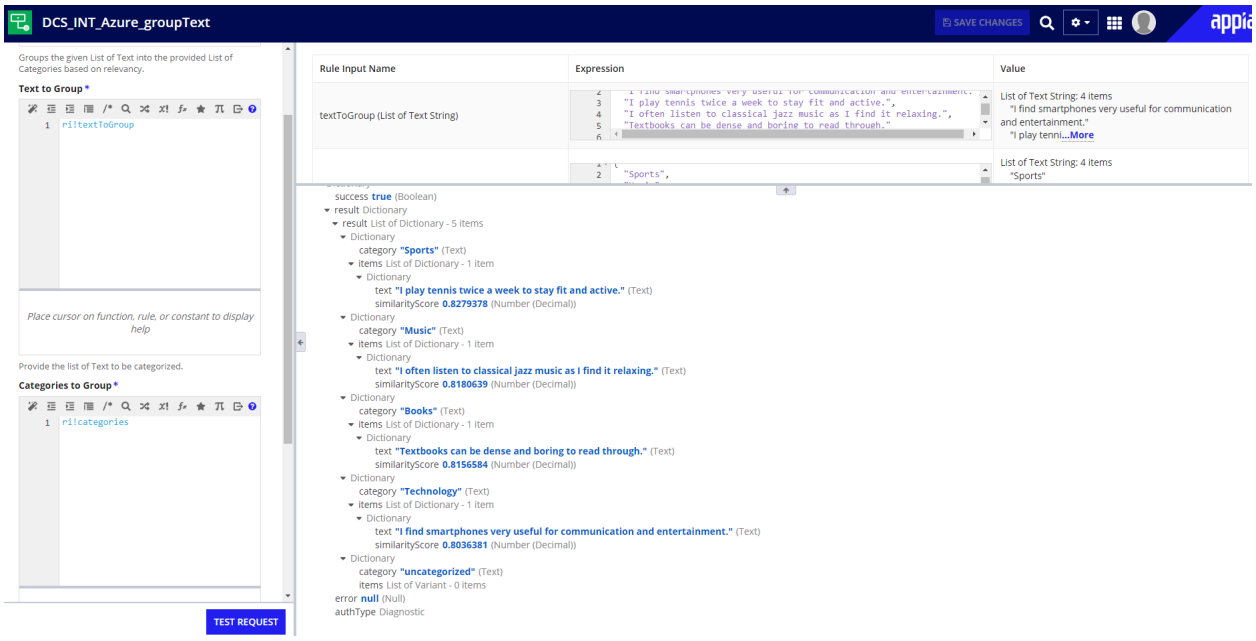
Is Unique(Boolean) - Optional

Description: Provide true to avoid duplication of items among the categories.

Minimum Similarity Score(Number(Decimal)) - Optional.

Description: Provide the minimum relevancy score for the text to be matched with the categories. The higher the minimum similarity score, more relevant texts will be grouped.Accepts values between 0 and 1. Defaults to 0.75.

Output:



Token Sizes for Completion Models

Model	Tokens
gpt-3.5-turbo	4,096
gpt-3.5-turbo-16k	16,385
gpt-4	8,192
gpt-4-32k	32,768

Reference Links:

- https://platform.openai.com/docs/models
- https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/models

Top K Configuration Guide

Refer the following table for the topK value configuration depending on the model you are using:

Model	Max Tokens for the model	Default Value	Maximum Allowed Value
-	4096	8	8
gpt-3.5-turbo	16385	24	41
gpt-4	8192	16	20
-	32768	32	95
gpt-4-turbo gpt-4-turbo-preview gpt-4o	128000	40	410

Query Behavior

When querying for a document, an initial slight delay in querying speed may be experienced as the document is processed and added to the cache. Following this initial query, the document remains in the cache for a duration of 10 minutes since its last usage. Subsequent queries on documents present in the cache will exhibit significantly faster response times, optimizing overall system performance.

Performance Metrics for Uploading Document

The below metrics are for reference on the performance of the Upload Documents Integration.

Total Number of Pages (approx.)	Time taken for Uploading (in ms)
100	6,702
200	11,947
500	31,284
1000	57,327

Performance Metrics for Query Documents

The below metrics are for reference on the performance of the Query Documents Integration.

Note: By using a caching mechanism to securely store and quickly retrieve document information, we have greatly reduced the time required for each document query. This improvement can be seen in consecutive queries to the same document. The initial query to a document will be a non-cached query; any subsequent queries to the same document will experience the faster times of a cached query, as described in the chart below.

Total Number of Pages (approx.)	**Time taken for Query Response (in ms)**	
	Non Cached	**Cached**
100	1,832	774
200	1,862	536
500	4,732	766
1000	5,988	819

Heap Usage Metrics for Querying Documents

The following metrics provide a reference for heap memory utilization during document querying. When selecting documents, consider the approximate number of chunks they contain. Refer to the Production Usage section for how to optimize performance for multiple concurrent users or a large number of documents.

Pages	Chunks(Approx.)	Minimum Heap Memory required (In MB)(Approx.)
500	2000	250
1000	4000	525
2000	8000	1050
3000	12000	1500

**The number of document chunks varies by the amount of documents queried, the number of pages in each document, and quantity of text on each page.

AI Knowledge Assistant Component Version Compatibility

The following table shows the compatible versions of AI Knowledge Assistant Component for each Document Vector Database Connected System version.

Sl. No	Connected System Version	Compatible Component Versions
1	1.0.0	2.0.0
2	1.0.1	2.0.0
3	1.0.2	2.0.0
4	1.0.3	2.0.0
5	1.0.4	2.0.0
6	2.0.0	2.1.0 2.1.1
7	2.0.1	2.1.0 2.1.1
8	3.0.0	3.0.0 3.0.1
9	3.0.1	3.0.0 3.0.1
10	3.1.0	3.0.2 3.0.3 3.0.4
11	3.1.1	3.0.2 3.0.3 3.0.4
12	3.1.2	3.0.2 3.0.3 3.0.4

13	3.1.3	3.0.2 3.0.3 3.0.4
14	3.2.0	3.1.0 3.1.1 3.1.2 3.1.3
15	3.2.1	3.1.0 3.1.1 3.1.2 3.1.3
16	3.2.2	3.1.0 3.1.1 3.1.2 3.1.3
17	3.2.3	3.1.4 3.1.5 3.1.6
18	3.2.4	3.1.4 3.1.5 3.1.6
19	3.2.5	3.1.4 3.1.5 3.1.6
20	3.2.6	3.1.4 3.1.5 3.1.6

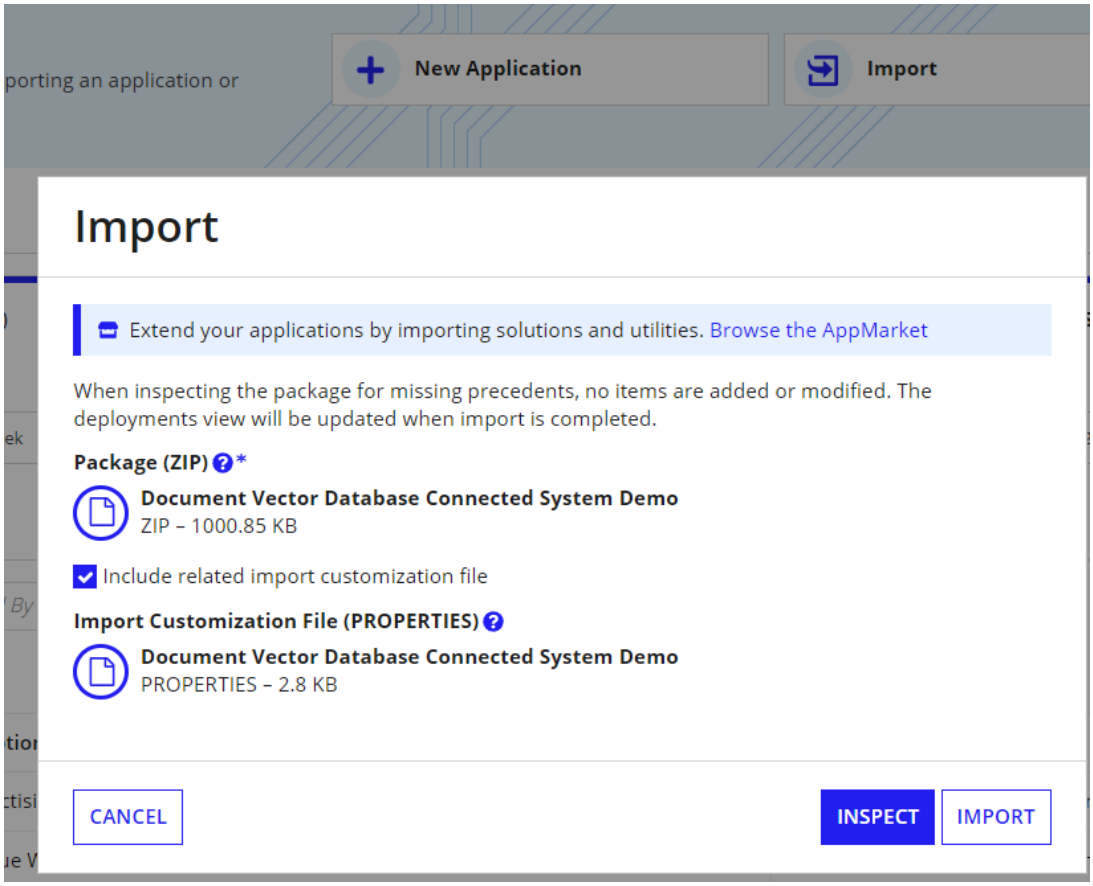
Document Vector Database Sample App Setup

This will walk you through importing the sample application

1. Open the "Document Vector Database Connected System Demo.properties" file in a text editor.

```
## Connected System: DCS Azure OpenAI Service CS
connectedSystem._a-0000ea91-9100-8000-62e3-01ef9001ef90_4481180.azureAPIKey=<--Provide your Azure API Key here-->
connectedSystem._a-0000ea91-9100-8000-62e3-01ef9001ef90_4481180.databasePassword=<--Provide your Database password here-->

## Connected System: DCS Azure OpenAI Service CS V2
connectedSystem._a-0000eb1a-5143-8000-630c-01ef9001ef90_4653548.azureAPIKey=<--Provide your Azure API Key here-->
connectedSystem._a-0000eb1a-5143-8000-630c-01ef9001ef90_4653548.databasePassword=<--Provide your Database password here-->

## Connected System: DCS Azure OpenAI Service CS V3
connectedSystem._a-0000eb53-1f71-8000-6318-01ef9001ef90_4698227.azureRegion=<--Provide your Azure region here-->
connectedSystem._a-0000eb53-1f71-8000-6318-01ef9001ef90_4698227.embeddingDeploymentId=<--Provide your Azure Deployment Id for embedding here-->
connectedSystem._a-0000eb53-1f71-8000-6318-01ef9001ef90_4698227.azureAPIKey=<--Provide your Azure API Key here-->
connectedSystem._a-0000eb53-1f71-8000-6318-01ef9001ef90_4698227.completionDeploymentId=<--Provide your Azure Deployment Id for completion here-->
connectedSystem._a-0000eb53-1f71-8000-6318-01ef9001ef90_4698227.maxTokens=<--Provide one of the following values here for Maximum tokens: 4096, 8192, 16384, 32768-->
connectedSystem._a-0000eb53-1f71-8000-6318-01ef9001ef90_4698227.databaseDocumentConstantName=<--Provide your Database name here-->
connectedSystem._a-0000eb53-1f71-8000-6318-01ef9001ef90_4698227.username=<--Provide your Appian Username here-->
connectedSystem._a-0000eb53-1f71-8000-6318-01ef9001ef90_4698227.databasePassword=<--Provide your Database password here-->

## Connected System: DCS OpenAI Service CS
connectedSystem._a-0000ea91-9100-8000-62e3-01ef9001ef90_4480987.openaiApiKey=<--Provide your OpenAI API Key here-->
connectedSystem._a-0000ea91-9100-8000-62e3-01ef9001ef90_4480987.databasePassword=<--Provide your Database password here-->

## Connected System: DCS OpenAI Service CS V2
connectedSystem._a-0000eb12-c62d-8000-6302-01ef9001ef90_4645306.openaiApiKey=<--Provide your OpenAI API Key here-->
connectedSystem._a-0000eb12-c62d-8000-6302-01ef9001ef90_4645306.databasePassword=<--Provide your Database password here-->

## Connected System: DCS OpenAI Service CS V3
connectedSystem._a-0000eb53-0529-8000-6317-01ef9001ef90_4697413.openaiApiKey=<--Provide your OpenAI API Key here-->
connectedSystem._a-0000eb53-0529-8000-6317-01ef9001ef90_4697413.openAIEmbeddingModel=<--Provide your OpenAI embedding model here-->
connectedSystem._a-0000eb53-0529-8000-6317-01ef9001ef90_4697413.completionModel=<--Provide your OpenAI completion model here-->
connectedSystem._a-0000eb53-0529-8000-6317-01ef9001ef90_4697413.databaseDocumentConstantName=<--Provide your Database name here-->
connectedSystem._a-0000eb53-0529-8000-6317-01ef9001ef90_4697413.username=<--Provide your Appian Username here-->
connectedSystem._a-0000eb53-0529-8000-6317-01ef9001ef90_4697413.databasePassword=<--Provide your Database password here-->
```

2. Replace the placeholder text with your Azure and OpenAI API keys for both Versions. For Database name and Database password, provide a name and password that will be used as the name and password for H2 Database. Provide suitable values for other fields. Save the file.
3. Import the "Document Vector Database Connected System Demo.zip" file into your Appian environment. Check on the Include related customization file check box and upload the properties file saved in the previous step.



4. Navigate to the "Document Vector Database Demo" site, click the site link, and test out the Document and Database operations through the menu.