

Starting April 29, 2025, Gemini 1.5 Pro and Gemini 1.5 Flash models are not available in projects that have no prior usage of these models, including new projects. For details, see [Model versions and lifecycle](#) (/vertex-ai/generative-ai/docs/learn/model-versions#legacy-stable).

RAG Engine API

The Vertex AI RAG Engine is a component of the Vertex AI platform, which facilitates Retrieval-Augmented Generation (RAG). RAG Engine enables Large Language Models (LLMs) to access and incorporate data from external knowledge sources, such as documents and databases. By using RAG, LLMs can generate more accurate and informative LLM responses.

Parameters list

This section lists the following:

Parameters	Examples
See Corpus management parameters (#corpus-management-params-api).	See Corpus management examples (#corpus-management-examples-api).
See File management parameters (#file-management-params-api).	See File management examples (#file-management-examples-api).

Corpus management parameters

For information about a RAG corpus, see [Corpus management](#) (/vertex-ai/generative-ai/docs/manage-your-rag-corpus#corpus-management).

Create a RAG corpus

This table lists the parameters used to create a RAG corpus.

Body Request

Parameters

display_name	Required: string The display name of the RAG corpus.
description	Optional: string The description of the RAG corpus.
vector_db_config	Optional: Immutable: RagVectorDbConfig The configuration for the Vector DBs.
vertex_ai_search_config.serving_config	Optional: string The configuration for the Vertex AI Search. Format: projects/{project}/locations/{location}/collections/{collection}/engines/{engine}/servingConfigs/{serving_config} or projects/{project}/locations/{location}/collections/{collection}/dataStores/{data_store}/servingConfigs/{serving_config}

RagVectorDbConfig

Parameters

rag_managed_db	oneof vector_db: RagVectorDbConfig.RagManagedDb If no vector database is specified, rag_managed_db is the default vector database.
pinecone	oneof vector_db: RagVectorDbConfig.Pinecone Specifies your Pinecone instance.
pinecone.index_name	string This is the name used to create the Pinecone index that's used with the RAG corpus. This value can't be changed after it's set. You can leave it empty in the CreateRagCorpus API call, and set it with a non-empty value in a follow up UpdateRagCorpus API call.
vertex_vector_search	oneof vector_db: RagVectorDbConfig.VertexVectorSearch

Specifies your Vertex Vector Search instance.

vertex_vector_search.index

string

This is the resource name of the Vector Search index that's used with the RAG corpus.

Format:

projects/{project}/locations/{location}/indexEndpoints/{index_endpoint}

This value can't be changed after it's set. You can leave it empty in the **CreateRagCorpus** API call, and set it with a non-empty value in a follow up **UpdateRagCorpus** API call.

vertex_vector_search.index_endpoint
int

This is the resource name of the Vector Search index endpoint that's used with the RAG corpus.

Format:

projects/{project}/locations/{location}/indexes/{index}

This value can't be changed after it's set. You can leave it empty in the **CreateRagCorpus** API call, and set it with a non-empty value in a follow up **UpdateRagCorpus** API call.

api_auth.api_key_config.api_key_string
secret_version

This the full resource name of the secret that is stored in Secret Manager, which contains your Pinecone API key.

Format:

projects/{PROJECT_NUMBER}/secrets/{SECRET_ID}/versions/{VERSION_ID}

You can leave it empty in the **CreateRagCorpus** API call, and set it with a non-empty value in a follow up **UpdateRagCorpus** API call.

rag_embedding_model_config.vertex_prediction_endpoint
Optional: Immutable: string

The embedding model to use for the RAG corpus. This value can't be changed after it's set. If you leave it empty, we use [text-embedding-005](https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/text-embeddings-api) (/vertex-ai/generative-ai/docs/model-reference/text-embeddings-api) as the embedding model.

Update a RAG corpus

This table lists the parameters used to update a RAG corpus.

Body Request

Parameters	
display_name	<div>Optional: string</div> <div>The display name of the RAG corpus.</div>
description	<div>Optional: string</div> <div>The description of the RAG corpus.</div>
rag_vector_db.pinecone.index_name	<div>string</div> <div>This is the name used to create the Pinecone index that's used with the RAG corpus.</div> <div>If your RagCorpus was created with a Pinecone configuration, and this field has never been set before, then you can update the Pinecone instance's index name.</div>
rag_vector_db.vertex_vector_search.index	<div>string</div> <div>This is the resource name of the Vector Search index that's used with the RAG corpus.</div> <div>Format: projects/{project}/locations/{location}/indexEndpoints/{index_endpoint}</div> <div>If your RagCorpus was created with a Vector Search configuration, and this field has never been set before, then you can update it.</div>
rag_vector_db.vertex_vector_search.index_endpoint	<div>string</div> <div>This is the resource name of the Vector Search index endpoint that's used with the RAG corpus.</div> <div>Format: projects/{project}/locations/{location}/indexes/{index}</div> <div>If your RagCorpus was created with a Vector Search configuration, and this field has never been set before, then you can update it.</div>

<code>rag_vector_db.api_auth.api_key_config.api_key_secret_version</code>	<div>The full resource name of the secret that is stored in Secret Manager, which contains your Pinecone API key.</div> <div>Format: <code>projects/{PROJECT_NUMBER}/secrets/{SECRET_ID}/versions/{VERSION_ID}</code></div>
---	---

List RAG corpora

This table lists the parameters used to list RAG corpora.

Parameters

<code>page_size</code>	<div>Optional: <code>int</code></div> <div>The standard list page size.</div>
<code>page_token</code>	<div>Optional: <code>string</code></div> <div>The standard list page token. Typically obtained from <code>[ListRagCorporaResponse.next_page_token][]</code> of the previous <code>[VertexRagDataService.ListRagCorpora][]</code> call.</div>

Get a RAG corpus

This table lists parameters used to get a RAG corpus.

Parameters

<code>name</code>	<div><code>string</code></div> <div>The name of the <code>RagCorpus</code> resource. Format: <code>projects/{project}/locations/{location}/ragCorpora/{rag_corpus_id}</code></div>
-------------------	--

Delete a RAG corpus

This table lists parameters used to delete a RAG corpus.

Parameters

name	string
	The name of the RagCorpus resource. Format: projects/{project}/locations/{location}/ragCorpora/{rag_corpus_id}

File management parameters

For information about a RAG file, see [File management](#)
(/vertex-ai/generative-ai/docs/manage-your-rag-corpus#file-management).

Upload a RAG file

This table lists parameters used to upload a RAG file.

Body Request

Parameters	
parent	string
	The name of the RagCorpus resource. Format: projects/{project}/locations/{location}/ragCorpora/{rag_corpus_id}
rag_file	Required: RagFile
	The file to upload.
upload_rag_file_config	Required: UploadRagFileConfig
	The configuration for the RagFile to be uploaded into the RagCorpus .
RagFile	
display_name	Required: string
	The display name of the RAG file.
description	Optional: string
	The description of the RAG file.

UploadRagFileConfig

<code>rag_file_transformation_config.rag_file_chunking_config.fixed_length_chunking.chunk_size</code>	Number of tokens each chunk has.
<code>rag_file_transformation_config.rag_file_chunking_config.fixed_length_chunking.chunk_overlap</code>	The overlap between chunks.

Import RAG files

This table lists parameters used to import a RAG file.

Parameters	
<code>parent</code>	<p>Required: <code>string</code></p> <p>The name of the <code>RagCorpus</code> resource.</p> <p>Format: <code>projects/{project}/locations/{location}/ragCorpora/{rag_corpus_id}</code></p>
<code>gcs_source</code>	<p><code>oneof import_source: GcsSource</code></p> <p>Cloud Storage location.</p> <p>Supports importing individual files as well as entire Cloud Storage directories.</p>
<code>gcs_source.uris</code>	<p><code>list of string</code></p> <p>Cloud Storage URI that contains the upload file.</p>
<code>google_drive_source</code>	<p><code>oneof import_source: GoogleDriveSource</code></p> <p>Google Drive location.</p> <p>Supports importing individual files as well as Google Drive folders.</p>
<code>slack_source</code>	<p><code>oneof import_source: SlackSource</code></p> <p>The slack channel where the file is uploaded.</p>

jira_source	oneof import_source: JiraSource The Jira query where the file is uploaded.
share_point_sources	oneof import_source: SharePointSources The SharePoint sources where the file is uploaded.
rag_file_transformation_config.rag_file_chunking_config.fixed_length_chunking.chunk_size	int32 Number of tokens each chunk has.
rag_file_transformation_config.rag_file_chunking_config.fixed_length_chunking.chunk_overlap	int32 The overlap between chunks.
rag_file_parsing_config	Optional: RagFileParsingConfig Specifies the parsing configuration for RagFiles. If this field isn't set, RAG uses the default parser.
max_embedding_requests_per_min	Optional: int32 The maximum number of queries per minute that this job is allowed to make to the embedding model specified on the corpus. This value is specific to this job and not shared across other import jobs. Consult the Quotas page on the project to set an appropriate value. If unspecified, a default value of 1,000 QPM is used.
GoogleDriveSource	
resource_ids.resource_id	Required: string The ID of the Google Drive resource.
resource_ids.resource_type	Required: string The type of the Google Drive resource.
SlackSource	
channels.channels	Repeated: SlackSource.SlackChannels.SlackChannel Slack channel information, include ID and time range to import.

<code>channels.channels.channel_id</code>	Required: string The Slack channel ID.
<code>channels.channels.start_time</code>	Optional: google.protobuf.Timestamp The starting timestamp for messages to import.
<code>channels.channels.end_time</code>	Optional: google.protobuf.Timestamp The ending timestamp for messages to import.
<code>channels.api_key_config.api_key_secret_version</code>	Required: string The full resource name of the secret that is stored in Secret Manager, which contains a Slack channel access token that has access to the slack channel IDs. See: https://api.slack.com/tutorials/tracks/getting-a-token . Format: <code>projects/{PROJECT_NUMBER}/secrets/{SECRET_ID}/versions/{VERSION_ID}</code>
JiraSource	
<code>jira_queries.projects</code>	Repeated: string A list of Jira projects to import in their entirety.
<code>jira_queries.custom_queries</code>	Repeated: string A list of custom Jira queries to import. For information about JQL (Jira Query Language), see Jira Support (https://support.atlassian.com/jira-service-management-cloud/docs/use-advanced-search-with-jira-query-language-jql/)
<code>jira_queries.email</code>	Required: string The Jira email address.
<code>jira_queries.server_uri</code>	Required: string The Jira server URI.
<code>jira_queries.api_key_config.api_key_secret_version</code>	Required: string The full resource name of the secret that is stored in Secret Manager, which contains Jira API key that has access to the slack channel IDs.

See: <https://support.atlassian.com/atlassian-account/docs/manage-api-tokens-for-your-atlassian-account/>

Format:
`projects/{PROJECT_NUMBER}/secrets/{SECRET_ID}/versions/{VERSION_ID}`

SharePointSources

<code>share_point_sources.sharepoint_foldoneof in folder_source: string</code> <code>er_path</code>	The path of the SharePoint folder to download from.
<code>share_point_sources.sharepoint_foldoneof in folder_source: string</code> <code>er_id</code>	The ID of the SharePoint folder to download from.
<code>share_point_sources.drive_name</code>	<code>oneof in drive_source: string</code> The name of the drive to download from.
<code>share_point_sources.drive_id</code>	<code>oneof in drive_source: string</code> The ID of the drive to download from.
<code>share_point_sources.client_id</code>	<code>string</code> The Application ID for the app registered in Microsoft Azure Portal. The application must also be configured with MS Graph permissions "Files.ReadAll", "Sites.ReadAll" and BrowserSiteLists.Read.All.
<code>share_point_sources.client_secret.aRequired: string</code> <code>pi_key_secret_version</code>	The full resource name of the secret that is stored in Secret Manager, which contains the application secret for the app registered in Azure. Format: <code>projects/{PROJECT_NUMBER}/secrets/{SECRET_ID}/versions/{VERSION_ID}</code>
<code>share_point_sources.tenant_id</code>	<code>string</code> Unique identifier of the Azure Active Directory Instance.
<code>share_point_sources.sharepoint_sitestring</code> <code>_name</code>	The name of the SharePoint site to download from. This can be the site name or the site id.

RagFileParsingConfig

layout_parser oneof parser: RagFileParsingConfig.LayoutParser

The Layout Parser to use for RagFiles.

layout_parser.processor_namestring

The full resource name of a Document AI processor or processor version.

Format:

projects/{project_id}/locations/{location}/processors/{processor_id}
projects/{project_id}/locations/{location}/processors/{processor_id}/processorVersions/{processor_version_id}

**layout_parser.max_parsing_restring
quests_per_min**

The maximum number of requests the job is allowed to make to the Document AI processor per minute.

Consult <https://cloud.google.com/document-ai/quotas> and the Quota page for your project to set an appropriate value here. If unspecified, a default value of 120 QPM is used.

llm_parser oneof parser: RagFileParsingConfig.LlmParser

The LLM parser to use for RagFiles.

llm_parser.model_name

string

The resource name of an LLM model.

Format:

projects/{project_id}/locations/{location}/publishers/{publisher}/models/{model}

**llm_parser.max_parsing_requeststring
sts_per_min**

The maximum number of requests the job is allowed to make to the LLM model per minute.

To set an appropriate value for your project, see [model quota section](#) (/vertex-ai/generative-ai/docs/quotas#quota_system_by_model) and the Quota page for your project to set an appropriate value here. If unspecified, a default value of 5000 QPM is used.

Get a RAG file

This table lists parameters used to get a RAG file.

Parameters	
name	<div>string</div> <div>The name of the RagFile resource. Format: projects/{project}/locations/{location}/ragCorpora/{rag_file_id}</div>

Delete a RAG file

This table lists parameters used to delete a RAG file.

Parameters	
name	<div>string</div> <div>The name of the RagFile resource. Format: projects/{project}/locations/{location}/ragCorpora/{rag_file_id}</div>

Retrieval and prediction

This section lists the retrieval and prediction parameters.

Retrieval parameters

This table lists parameters for `retrieveContexts` API.

Parameters	
parent	<div>Required: string</div> <div>The resource name of the Location to retrieve RagContexts. The users must have permission to make a call in the project.</div> <div>Format: projects/{project}/locations/{location}</div>
vertex_rag_store	<div>VertexRagStore</div> <div>The data source for Vertex RagStore.</div>

query	Required: RagQuery
	Single RAG retrieve query.
VertexRagStore	
VertexRagStore	
rag_resources	list: RagResource
	The representation of the RAG source. It can be used to specify the corpus only or RagFiles. Only support one corpus or multiple files from one corpus.
rag_resources.rag_corpus	Optional: string
	RagCorpora resource name.
	Format: projects/{project}/locations/{location}/ragCorpora/{rag_corpus}
rag_resources.rag_file_ids	list: string
	A list of RagFile resources.
	Format: projects/{project}/locations/{location}/ragCorpora/{rag_corpus}/ragFiles/{rag_file}
RagQuery	
text	string
	The query in text format to get relevant contexts.
rag_retrieval_config	Optional: RagRetrievalConfig
	The retrieval configuration for the query.
RagRetrievalConfig	
top_k	Optional: int32
	The number of contexts to retrieve.
filter.vector_distance_threshold	oneof vector_db_threshold: double

Only returns contexts with a vector distance smaller than the threshold.

<code>filter.vector_similarity_threshold</code>	Optional: <code>double</code> Only returns contexts with vector similarity larger than the threshold.
<code>ranking.rank_service.model_name</code>	Optional: <code>string</code> The model name of the rank service. Example: <code>semantic-ranker-512@latest</code>
<code>ranking.llm_ranker.model_name</code>	Optional: <code>string</code> The model name used for ranking. Example: <code>gemini-2.0-flash</code>

Prediction parameters

This table lists prediction parameters.

GenerateContentRequest

<code>tools.retrieval.vertex_rag_store</code>	<code>VertexRagStore</code> Set to use a data source powered by Vertex AI RAG store.
---	---

See [VertexRagStore](#) (`#vertex-rag-store`) for details.

Corpus management examples

This section provides examples of how to use the API to manage your RAG corpus.

Create a RAG corpus example

These code samples demonstrate how to create a RAG corpus.

`RESTPython` (`#python`)
`(#rest)`

Before using any of the request data, make the following replacements:

- **PROJECT_ID**: Your project ID.
- **LOCATION**: The region to process the request.
- **CORPUS_DISPLAY_NAME**: The display name of the RAG corpus.
- **CORPUS_DESCRIPTION**: The description of the RAG corpus.

HTTP method and URL:

POST https://LOCATION-aiplatform.googleapis.com/v1/projects/PROJECT_ID/loc

Request JSON body:

```
{
  "display_name" : "CORPUS_DISPLAY_NAME",
  "description": "CORPUS_DESCRIPTION",
}
```

To send your request, choose one of these options:

curl**Powershell** (#powershell)
(#curl)

★ **Note:** The following command assumes that you have signed in to the Google Cloud CLI CLI with your user account by running `gcloud CLI init` or `gcloud CLI auth login`, or by using Cloud Shell, which automatically signs you into the `gcloud CLI CLI`. You can check the active account by running `gcloud CLI auth list`.

Save the request body in a file named `request.json`, and run the following command:

```
curl -X POST \
  -H "Authorization: Bearer $(gcloud auth print-access-token)" \
  -H "Content-Type: application/json; charset=utf-8" \
  -d @request.json \
  "https://LOCATION-aiplatform.googleapis.com/v1/projects/PROJECT_ID"
```

You should receive a successful status code (2xx).

The following example demonstrates how to create a RAG corpus by using the REST API.

```
// CreateRagCorpus
// Input: LOCATION, PROJECT_ID, CORPUS_DISPLAY_NAME
// Output: CreateRagCorpusOperationMetadata
curl -X POST \
-H "Authorization: Bearer $(gcloud auth print-access-token)" \
-H "Content-Type: application/json" \
https://LOCATION-aiplatform.googleapis.com/v1/projects/PROJECT_ID/locations/LOCATION \
-d '{
  "display_name" : "CORPUS_DISPLAY_NAME"
}'
```

Update a RAG corpus example

You can update your RAG corpus with a new display name, description, and vector database configuration. However, you can't change the following parameters (#update-a-rag-corpus-params-api) in your RAG corpus:

- The vector database type. For example, you can't change the vector database from Weaviate to Vertex AI Feature Store.
- If you're using the managed database option, you can't update the vector database configuration.

These examples demonstrate how to update a RAG corpus.

REST (#rest)

Before using any of the request data, make the following replacements:

- ***PROJECT_ID***: Your project ID.
- ***LOCATION***: The region to process the request.
- ***CORPUS_ID***: The corpus ID of your RAG corpus.
- ***CORPUS_DISPLAY_NAME***: The display name of the RAG corpus.
- ***CORPUS_DESCRIPTION***: The description of the RAG corpus.

- **INDEX_NAME**: The resource name of the Vector Search Index. Format:
projects/{project}/locations/{location}/indexes/{index}.
- **INDEX_ENDPOINT_NAME**: The resource name of the Vector Search index endpoint.
Format:
projects/{project}/locations/{location}/indexEndpoints/{index_endpoint}.

HTTP method and URL:

PATCH https://LOCATION -aiplatform.googleapis.com/v1/projects/PROJECT_ID/lc

Request JSON body:

```
{
  "display_name" : "CORPUS_DISPLAY_NAME",
  "description": "CORPUS_DESCRIPTION",
  "rag_vector_db_config": {
    "vertex_vector_search": {
      "index": "INDEX_NAME",
      "index_endpoint": "INDEX_ENDPOINT_NAME",
    }
  }
}
```

To send your request, choose one of these options:

curl**Powershell** (#powershell)
(#curl)

★ **Note:** The following command assumes that you have signed in to the Google Cloud CLI CLI with your user account by running gcloud CLI **init** or gcloud CLI **auth login**, or by using Cloud Shell, which automatically signs you into the gcloud CLI CLI . You can check the active account by running gcloud CLI **auth list**.

Save the request body in a file named request.json, and run the following command:

```
curl -X PATCH \
  -H "Authorization: Bearer $(gcloud auth print-access-token)" \
  -H "Content-Type: application/json; charset=utf-8" \
  -d @request.json \
  "https://LOCATION -aiplatform.googleapis.com/v1/projects/PROJECT_ID
```

You should receive a successful status code (2xx).

List RAG corpora example

These code samples demonstrate how to list all of the RAG corpora.

RESTPython (#python)
(#rest)

Before using any of the request data, make the following replacements:

- ***PROJECT_ID***: Your project ID.
- ***LOCATION***: The region to process the request.
- ***PAGE_SIZE***: The standard list page size. You might adjust the number of RAG corpora to return per page by updating the `page_size` parameter.
- ***PAGE_TOKEN***: The standard list page token. Obtained typically using `ListRagCorporaResponse.next_page_token` of the previous `VertexRagDataService.ListRagCorpora` call.

HTTP method and URL:

GET `https://LOCATION -aiplatform.googleapis.com/v1/projects/PROJECT_ID /loc`

To send your request, choose one of these options:

curlPowershell (#powershell)
(#curl)

★ **Note:** The following command assumes that you have signed in to the Google Cloud CLI CLI with your user account by running `gcloud CLI init` or `gcloud CLI auth login`, or by using Cloud Shell,

which automatically signs you into the gcloud CLI CLI . You can check the active account by running gcloud CLI **auth list**.

Run the following command:

```
curl -X GET \
  -H "Authorization: Bearer $(gcloud auth print-access-token)" \
  "https://LOCATION -aiplatform.googleapis.com/v1/projects/PROJECT_ID
```

You should receive a successful status code (2xx) and a list of RAG corpora under the given **PROJECT_ID**.

Get a RAG corpus example

These code samples demonstrate how to get a RAG corpus.

RESTPython (#python)
(#rest)

Before using any of the request data, make the following replacements:

- **PROJECT_ID**: Your project ID.
- **LOCATION**: The region to process the request.
- **RAG_CORPUS_ID**: The ID of the RAG corpus resource.

HTTP method and URL:

GET [https://LOCATION -aiplatform.googleapis.com/v1/projects/PROJECT_ID /loc](https://LOCATION -aiplatform.googleapis.com/v1/projects/PROJECT_ID/loc)

To send your request, choose one of these options:

curlPowershell (#powershell)
(#curl)

★ **Note:** The following command assumes that you have signed in to the Google Cloud CLI CLI with your user account by running gcloud CLI **init** or gcloud CLI **auth login**, or by using Cloud Shell,

which automatically signs you into the gcloud CLI . You can check the active account by running gcloud CLI **auth list**.

Run the following command:

```
curl -X GET \
  -H "Authorization: Bearer $(gcloud auth print-access-token)" \
  "https://LOCATION -aiplatform.googleapis.com/v1/projects/PROJECT_ID
```

A successful response returns the RagCorpus resource.

The **get** and **list** commands are used in an example to demonstrate how RagCorpus uses the **rag_embedding_model_config** field with in the **vector_db_config**, which points to the embedding model you have chosen.

PROJECT_ID : Your project ID.

LOCATION : The region to process the request.

RAG_CORPUS_ID : The corpus ID of your RAG corpus.

```
...
```sh
// GetRagCorpus
// Input: LOCATION, PROJECT_ID, RAG_CORPUS_ID
// Output: RagCorpus
curl -X GET \
 -H "Content-Type: application/json" \
 -H "Authorization: Bearer $(gcloud auth print-access-token)" \
 https://LOCATION -aiplatform.googleapis.com/v1/projects/PROJECT_ID /locati

// ListRagCorpora
curl -sS -X GET \
 -H "Content-Type: application/json" \
 -H "Authorization: Bearer $(gcloud auth print-access-token)" \
 https://LOCATION -aiplatform.googleapis.com/v1/projects/PROJECT_ID /locati
...

```

## Delete a RAG corpus example

These code samples demonstrate how to delete a RAG corpus.

**RESTPython** (#python)  
(#rest)

Before using any of the request data, make the following replacements:

- **PROJECT\_ID**: Your project ID.
- **LOCATION**: The region to process the request.
- **RAG\_CORPUS\_ID**: The ID of the RagCorpus resource.

HTTP method and URL:

DELETE [https://LOCATION-aiplatform.googleapis.com/v1/projects/PROJECT\\_ID/](https://LOCATION-aiplatform.googleapis.com/v1/projects/PROJECT_ID/)

To send your request, choose one of these options:

**curlPowershell** (#powershell)  
(#curl)

★ **Note:** The following command assumes that you have signed in to the Google Cloud CLI CLI with your user account by running `gcloud CLI init` or `gcloud CLI auth login`, or by using Cloud Shell, which automatically signs you into the gcloud CLI CLI . You can check the active account by running `gcloud CLI auth list`.

Run the following command:

```
curl -X DELETE \
 -H "Authorization: Bearer $(gcloud auth print-access-token)" \
 "https://LOCATION-aiplatform.googleapis.com/v1/projects/PROJECT_ID/
```

A successful response returns the `DeleteOperationMetadata`.

## File management examples

This section provides examples of how to use the API to manage RAG files.

## Upload a RAG file example

These code samples demonstrate how to upload a RAG file.

**RESTPython** (#python)  
(#rest)

Before using any of the request data, make the following replacements:

- **PROJECT\_ID**: Your project ID.
- **LOCATION**: The region to process the request.
- **RAG\_CORPUS\_ID**: The corpus ID of your RAG corpus.
- **LOCAL\_FILE\_PATH**: The local path to the file to be uploaded.
- **DISPLAY\_NAME**: The display name of the RAG file.
- **DESCRIPTION**: The description of the RAG file.

To send your request, use the following command:

```
curl -X POST \
-H "X-Goog-Upload-Protocol: multipart" \
-H "Authorization: Bearer $(gcloud auth print-access-token)" \
-F metadata="{ 'rag_file': { 'display_name': 'DISPLAY_NAME', 'description': 'DE' } }" \
-F file=@LOCAL_FILE_PATH \
"https://LOCATION-aiplatform.googleapis.com/upload/v1/projects/PROJECT_ID/"
```

## Import RAG files example

Files and folders can be imported from Drive or Cloud Storage. You can use `response.metadata` to view partial failures, request time, and response time in the SDK's `response` object.

The `response.skipped_rag_files_count` refers to the number of files that were skipped during import. A file is skipped when the following conditions are met:

1. The file has already been imported.
2. The file hasn't changed.

### 3. The chunking configuration for the file hasn't changed.

Python (#python)REST  
(#rest)

Before using any of the request data, make the following replacements:

- **PROJECT\_ID**: Your project ID.
- **LOCATION**: The region to process the request.
- **RAG\_CORPUS\_ID**: The corpus ID of your RAG corpus.
- **FOLDER\_RESOURCE\_ID**: The resource ID of your Drive folder.
- **GCS\_URIS**: A list of Cloud Storage locations. Example: gs://my-bucket1.
- **CHUNK\_SIZE**: Number of tokens each chunk should have.
- **CHUNK\_OVERLAP**: Number of tokens overlap between chunks.
- **EMBEDDING\_MODEL\_QPM\_RATE**: The QPM rate to limit RAG's access to your embedding model. Example: 1,000.

HTTP method and URL:

POST [https://LOCATION-aiplatform.googleapis.com/v1/projects/PROJECT\\_ID/loc](https://LOCATION-aiplatform.googleapis.com/v1/projects/PROJECT_ID/loc)

Request JSON body:


```
{
 "import_rag_files_config": {
 "gcs_source": {
 "uris": "GCS_URIS"
 },
 "rag_file_chunking_config": {
 "chunk_size": "CHUNK_SIZE",
 "chunk_overlap": "CHUNK_OVERLAP"
 }
 }
}
```

To send your request, choose one of these options:

curlPowershell (#powershell)  
(#curl)

★ **Note:** The following command assumes that you have signed in to the Google Cloud CLI CLI with your user account by running `gcloud CLI init` or `gcloud CLI auth login`, or by using Cloud Shell, which automatically signs you into the `gcloud CLI`. You can check the active account by running `gcloud CLI auth list`.

Save the request body in a file named `request.json`, and run the following command:

```
curl -X POST \
 -H "Authorization: Bearer $(gcloud auth print-access-token)" \
 -H "Content-Type: application/json; charset=utf-8" \
 -d @request.json \
 "https://LOCATION  -aiplatform.googleapis.com/v1/projects/PROJECT_ID
```

A successful response returns the `ImportRagFilesOperationMetadata` resource.

The following sample demonstrates how to import a file from Cloud Storage. Use the `max_embedding_requests_per_min` control field to limit the rate at which RAG Engine calls the embedding model during the `ImportRagFiles` indexing process. The field has a default value of 1000 calls per minute.

- **PROJECT\_ID:** Your project ID.
- **LOCATION:** The region to process the request.
- **RAG\_CORPUS\_ID:** The corpus ID of your RAG corpus.
- **GCS\_URIS:** A list of Cloud Storage locations. Example: `gs://my-bucket1`.
- **CHUNK\_SIZE:** Number of tokens each chunk should have.
- **CHUNK\_OVERLAP:** Number of tokens overlap between chunks.
- **EMBEDDING\_MODEL\_QPM\_RATE:** The QPM rate to limit RAGs access to your embedding model. Example: 1,000.



```
// ImportRagFiles
// Import a single Cloud Storage file or all files in a Cloud Storage bucket.
// Input: LOCATION, PROJECT_ID, RAG_CORPUS_ID, GCS_URIS
// Output: ImportRagFilesOperationMetadataNumber
// Use ListRagFiles to find the server-generated rag_file_id.
curl -X POST \
-H "Authorization: Bearer $(gcloud auth print-access-token)" \
-H "Content-Type: application/json" \
https://LOCATION -aiplatform.googleapis.com/v1/projects/PROJECT_ID/locations/LOCATION
-d '{
 "import_rag_files_config": {
 "gcs_source": {
 "uris": "GCS_URIS"
 },
 "rag_file_chunking_config": {
 "chunk_size": CHUNK_SIZE,
 "chunk_overlap": CHUNK_OVERLAP
 },
 "max_embedding_requests_per_min": EMBEDDING_MODEL_QPM_RATE
 }
}
```

The following sample demonstrates how to import a file from Drive. Use the `max_embedding_requests_per_min` control field to limit the rate at which RAG Engine calls the embedding model during the `ImportRagFiles` indexing process. The field has a default value of 1000 calls per minute.

- **PROJECT\_ID**: Your project ID.
- **LOCATION**: The region to process the request.
- **RAG\_CORPUS\_ID**: The corpus ID of your RAG corpus.
- **FOLDER\_RESOURCE\_ID**: The resource ID of your Drive folder.
- **CHUNK\_SIZE**: Number of tokens each chunk should have.
- **CHUNK\_OVERLAP**: Number of tokens overlap between chunks.
- **EMBEDDING\_MODEL\_QPM\_RATE**: The QPM rate to limit RAG's access to your embedding model. Example: 1,000.

```
// ImportRagFiles
// Import all files in a Google Drive folder.
// Input: LOCATION, PROJECT_ID, RAG_CORPUS_ID, FOLDER_RESOURCE_ID
// Output: ImportRagFilesOperationMetadataNumber
// Use ListRagFiles to find the server-generated rag_file_id.
curl -X POST \
-H "Authorization: Bearer $(gcloud auth print-access-token)" \
-H "Content-Type: application/json" \
https://LOCATION -aiplatform.googleapis.com/v1/projects/PROJECT_ID/location
-d '{
 "import_rag_files_config": {
 "google_drive_source": {
 "resource_ids": {
 "resource_id": "FOLDER_RESOURCE_ID",
 "resource_type": "RESOURCE_TYPE_FOLDER"
 }
 },
 "max_embedding_requests_per_min": EMBEDDING_MODEL_QPM_RATE
 }
}'
```

## List RAG files example

These code samples demonstrate how to list RAG files.

**RESTPython** (#python)  
(#rest)

Before using any of the request data, make the following replacements:

- **PROJECT\_ID**: Your project ID.
- **LOCATION**: The region to process the request.
- **RAG\_CORPUS\_ID**: The ID of the RagCorpus resource.
- **PAGE\_SIZE**: The standard list page size. You might adjust the number of RagFiles to return per page by updating the page\_size parameter.
- **PAGE\_TOKEN**: The standard list page token. Obtained using `ListRagFilesResponse.next_page_token` of the previous `VertexRagDataService.ListRagFiles` call.

## HTTP method and URL:

GET `https://LOCATION-aiplatform.googleapis.com/v1/projects/PROJECT_ID/loc`

To send your request, choose one of these options:

curlPowershell (#powershell)  
(#curl)

★ **Note:** The following command assumes that you have signed in to the Google Cloud CLI CLI with your user account by running `gcloud CLI init` or `gcloud CLI auth login`, or by using Cloud Shell, which automatically signs you into the `gcloud CLI` CLI . You can check the active account by running `gcloud CLI auth list`.

Run the following command:

```
curl -X GET \
 -H "Authorization: Bearer $(gcloud auth print-access-token)" \
 "https://LOCATION-aiplatform.googleapis.com/v1/projects/PROJECT_ID
```

You should receive a successful status code (2xx) along with a list of `RagFiles` under the given `RAG_CORPUS_ID`.

## Get a RAG file example

These code samples demonstrate how to get a RAG file.

RESTPython (#python)  
(#rest)

Before using any of the request data, make the following replacements:

- ***PROJECT\_ID***: Your project ID.
- ***LOCATION***: The region to process the request.
- ***RAG\_CORPUS\_ID***: The ID of the `RagCorpus` resource.

- **RAG\_FILE\_ID**: The ID of the RagFile resource.

HTTP method and URL:

GET [https://LOCATION-aiplatform.googleapis.com/v1/projects/PROJECT\\_ID/locations/RAG\\_ENGINE\\_INSTANCE\\_ID/ragFiles/RAG\\_FILE\\_ID](https://LOCATION-aiplatform.googleapis.com/v1/projects/PROJECT_ID/locations/RAG_ENGINE_INSTANCE_ID/ragFiles/RAG_FILE_ID)

To send your request, choose one of these options:

**curlPowershell** (#powershell)  
(#curl)

★ **Note:** The following command assumes that you have signed in to the Google Cloud CLI with your user account by running `gcloud CLI init` or `gcloud CLI auth login`, or by using Cloud Shell, which automatically signs you into the `gcloud CLI`. You can check the active account by running `gcloud CLI auth list`.

Run the following command:

```
curl -X GET \
 -H "Authorization: Bearer $(gcloud auth print-access-token)" \
 "https://LOCATION-aiplatform.googleapis.com/v1/projects/PROJECT_ID/locations/RAG_ENGINE_INSTANCE_ID/ragFiles/RAG_FILE_ID"
```

A successful response returns the RagFile resource.

## Delete a RAG file example

These code samples demonstrate how to delete a RAG file.

**RESTPython** (#python)  
(#rest)

Before using any of the request data, make the following replacements:

- **PROJECT\_ID**: Your project ID.
- **LOCATION**: The region to process the request.
- **RAG\_CORPUS\_ID**: The ID of the RagCorpus resource.

- **RAG\_FILE\_ID**: The ID of the RagFile resource. Format: `projects/{project}/locations/{location}/ragCorpora/{rag_corpus}/ragFiles/{rag_file_id}`.

HTTP method and URL:

DELETE `https://LOCATION -aiplatform.googleapis.com/v1/projects/PROJECT_ID /1`

To send your request, choose one of these options:

curlPowershell (#powershell)  
(#curl)

★ **Note:** The following command assumes that you have signed in to the Google Cloud CLI CLI with your user account by running `gcloud CLI init` or `gcloud CLI auth login`, or by using Cloud Shell, which automatically signs you into the `gcloud CLI CLI`. You can check the active account by running `gcloud CLI auth list`.

Run the following command:

```
curl -X DELETE \
 -H "Authorization: Bearer $(gcloud auth print-access-token)" \
 "https://LOCATION -aiplatform.googleapis.com/v1/projects/PROJECT_ID
```

## Retrieval query

When a user asks a question or provides a prompt, the retrieval component in RAG searches through its knowledge base to find information that is relevant to the query.

Python (#python)REST  
(#rest)

Before using any of the request data, make the following replacements:

- **PROJECT\_ID**: Your project ID.
- **LOCATION**: The region to process the request.

- **RAG\_CORPUS\_RESOURCE**: The name of the RagCorpus resource. Format: `projects/{project}/locations/{location}/ragCorpora/{rag_corpus}`.
- **VECTOR\_DISTANCE\_THRESHOLD**: Only contexts with a vector distance smaller than the threshold are returned.
- **TEXT**: The query text to get relevant contexts.
- **SIMILARITY\_TOP\_K**: The number of top contexts to retrieve.

HTTP method and URL:

POST `https://LOCATION -aiplatform.googleapis.com/v1/projects/PROJECT_ID/loc`

Request JSON body:

```
{
 "vertex_rag_store": {
 "rag_resources": {
 "rag_corpus": "RAG_CORPUS_RESOURCE"
 },
 "vector_distance_threshold": VECTOR_DISTANCE_THRESHOLD
 },
 "query": {
 "text": TEXT
 "similarity_top_k": SIMILARITY_TOP_K
 }
}
```

**curlPowershell** (#powershell)  
(#curl)

★ **Note:** The following command assumes that you have signed in to the Google Cloud CLI CLI with your user account by running `gcloud CLI init` or `gcloud CLI auth login`, or by using Cloud Shell, which automatically signs you into the `gcloud CLI CLI`. You can check the active account by running `gcloud CLI auth list`.

Save the request body in a file named `request.json`, and run the following command:

```
curl -X POST \
 -H "Authorization: Bearer $(gcloud auth print-access-token)" \
 -H "Content-Type: application/json; charset=utf-8" \
 -d @request.json \
 "https://LOCATION-aiplatform.googleapis.com/v1/projects/PROJECT_ID
```

You should receive a successful status code (2xx) and a list of related RagFiles.

## Generation

The LLM generates a grounded response using the retrieved contexts.

RESTPython (#python)  
(#rest)

Before using any of the request data, make the following replacements:

- ***PROJECT\_ID***: Your project ID.
- ***LOCATION***: The region to process the request.
- ***MODEL\_ID***: LLM model for content generation. Example: **gemini-2.0-flash-001**.
- ***GENERATION\_METHOD***: LLM method for content generation. Options: **generateContent**, **streamGenerateContent**.
- ***INPUT\_PROMPT***: The text sent to the LLM for content generation. Try to use a prompt relevant to the uploaded rag Files.
- ***RAG\_CORPUS\_RESOURCE***: The name of the RagCorpus resource. Format: **projects/{project}/locations/{location}/ragCorpora/{rag\_corpus}**.
- ***SIMILARITY\_TOP\_K***: Optional: The number of top contexts to retrieve.
- ***VECTOR\_DISTANCE\_THRESHOLD***: Optional: Contexts with a vector distance smaller than the threshold are returned.
- ***USER***: Your username.

HTTP method and URL:

POST **https://*LOCATION*-aiplatform.googleapis.com/v1/projects/*PROJECT\_ID*/loc**

Request JSON body:

```
{
 "contents": {
 "role": "USER ✎",
 "parts": {
 "text": "INPUT_PROMPT ✎"
 }
 },
 "tools": {
 "retrieval": {
 "disable_attribution": false,
 "vertex_rag_store": {
 "rag_resources": {
 "rag_corpus": "RAG_CORPUS_RESOURCE ✎"
 },
 "similarity_top_k": "SIMILARITY_TOP_K ✎",
 "vector_distance_threshold": VECTOR_DISTANCE_THRESHOLD ✎
 }
 }
 }
}
```

To send your request, choose one of these options:


curlPowershell (#powershell)  
(#curl)

★ **Note:** The following command assumes that you have signed in to the Google Cloud CLI CLI with your user account by running `gcloud CLI init` or `gcloud CLI auth login`, or by using Cloud Shell, which automatically signs you into the `gcloud CLI CLI`. You can check the active account by running `gcloud CLI auth list`.

Save the request body in a file named `request.json`, and execute the following command:

```
curl -X POST \
 -H "Authorization: Bearer $(gcloud auth print-access-token)" \
```



```
-H "Content-Type: application/json; charset=utf-8" \
-d @request.json \
"https://LOCATION  -aiplatform.googleapis.com/v1/projects/PROJECT_ID
```

A successful response returns the generated content with citations.

## What's next

- To learn more about supported generation models, see [Generative AI models that support RAG](https://vertex-ai/generative-ai/docs/supported-rag-models) (/vertex-ai/generative-ai/docs/supported-rag-models).
- To learn more about supported embedding models, see [Embedding models](https://vertex-ai/generative-ai/docs/use-embedding-models#supported-embedding-models) (/vertex-ai/generative-ai/docs/use-embedding-models#supported-embedding-models).
- To learn more about open models, see [Open models](https://vertex-ai/generative-ai/docs/use-embedding-models#use-oss-embedding-models) (/vertex-ai/generative-ai/docs/use-embedding-models#use-oss-embedding-models).
- To learn more about RAG Engine, see [RAG Engine overview](https://vertex-ai/generative-ai/docs/rag-overview) (/vertex-ai/generative-ai/docs/rag-overview).

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (https://creativecommons.org/licenses/by/4.0/), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (https://www.apache.org/licenses/LICENSE-2.0). For details, see the [Google Developers Site Policies](https://developers.google.com/site-policies) (https://developers.google.com/site-policies). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2025-06-06 UTC.