Submodules

genai.client module

class genai.client.**AsyncClient**(api_client)

Bases: `object`

Client for making asynchronous (non-blocking) requests.

property batches: AsyncBatches

property caches: AsyncCaches

property chats: AsyncChats

property files: AsyncFiles

property live: AsyncLive

property models: AsyncModels

property operations: AsyncOperations

property tunings: AsyncTunings

class genai.client.**Client**(*, vertexai=None, api_key=None, credentials=None, project=None, location=None, debug_config=None, http_options=None)

Skip to content

Bases: `object`

Client for making synchronous requests.

Use this client to make a request to the Gemini Developer API or Vertex AI API and then wait for the response.

To initialize the client, provide the required arguments either directly or by using environment variables. Gemini API users and Vertex AI users in express mode can provide API key by providing input argument *api_key="your-api-key"* or by defining *GOOGLE_API_KEY="your-api-key"* as an environment variable

Vertex AI API users can provide inputs argument as *vertexai=True, project="your-project-id", location="us-central1"* or by defining *GOOGLE_GENAI_USE_VERTEXAI=true*, *GOOGLE_CLOUD_PROJECT* and *GOOGLE_CLOUD_LOCATION* environment variables.

api_key

The API key to use for authentication. Applies to the Gemini Developer API only.

vertexai

Indicates whether the client should use the Vertex AI API endpoints. Defaults to False (uses Gemini Developer API endpoints). Applies to the Vertex AI API only.

credentials

The credentials to use for authentication when calling the Vertex AI APIs. Credentials can be obtained from environment variables and default credentials. For more information, see Set up Application Default Credentials. Applies to the Vertex AI API only.

project

The Google Cloud project ID to use for quota. Can be obtained from environment variables (for example, `GOOGLE_CLOUD_PROJECT`). Applies to the Vertex AI API only.

location

The location to send API requests to (for example, `us-central1`). Can be obtained from environment variables. Applies to the Vertex AI API only.

debug_config

Config settings that control network behavior of the client. This is typically used when running test code.

http_options

Http options to use for the client. These options will be applied to all requests made by the client. Example usage: *client = genai.Client(http_options=types.HttpOptions(api_version='v1'))*.

Usage for the Gemini Developer API:

```
e import genai

client = genai.Client(api_key='my-api-key')
```

Skip to content

Usage for the Vertex AI API:

```
from google import genai

client = genai.Client(
    vertexai=True, project='my-project-id', location='us-central1'
)
```

Initializes the client.

PARAMETERS:

- **vertexai** (*bool*) – Indicates whether the client should use the Vertex AI API endpoints. Defaults to False (uses Gemini Developer API endpoints). Applies to the Vertex AI API only.
- **api_key** (*str*) –

  The API key to use for authentication. Applies to the Gemini Developer API only.

- **credentials** (*google.auth.credentials.Credentials*) –

  The credentials to use for authentication when calling the Vertex AI APIs. Credentials can be obtained from environment variables and default credentials. For more information, see Set up Application Default Credentials. Applies to the Vertex AI API only.

- **project** (*str*) –

  The Google Cloud project ID to use for quota. Can be obtained from environment variables (for example, `GOOGLE_CLOUD_PROJECT`). Applies to the Vertex AI API only.

- **location** (*str*) –

  The location to send API requests to (for example, `us-central1`). Can be obtained from environment variables. Applies to the Vertex AI API only.

- **debug_config** (*DebugConfig*) – Config settings that control network behavior of the client. This is typically used when running test code.
- **http_options** (*Union[HttpOptions, HttpOptionsDict]*) – Http options to use for the client.

property aio: AsyncClient

property batches: Batches

property caches: Caches

property chats: Chats

property files: Files

property models: Models

property operations: Operations

property tunings: Tunings

property vertexai: bool

Returns whether the client is using the Vertex AI API.

Skip to content

genai.client.**DebugConfig**

Bases: `BaseModel`

Configuration options that change client network behavior when testing.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `client_mode (str | None)`
- `replay_id (str | None)`
- `replays_directory (str | None)`

field client_mode: Optional[str] [Optional]

field replay_id: Optional[str] [Optional]

field replays_directory: Optional[str] [Optional]

Skip to content

genai.batches module

class genai.batches.**AsyncBatches**(api_client_)

Skip to content

Bases: `BaseModule`

async cancel(*, name, config=None)

Cancels a batch job.

Only available for batch jobs that are running or pending.

RETURN TYPE:

`None`

PARAMETERS:

name (*str*) –

A fully-qualified BatchJob resource name or ID. Example: “projects/…/locations/…/batchPredictionJobs/123456789” or

> ”123456789” when project and location are initialized in the client.

Usage:

```
await client.aio.batches.cancel(name='123456789')
```

async create(*, model, src, config=None)

Creates a batch job asynchronously.

RETURN TYPE:

`BatchJob`

PARAMETERS:

- **model** (*str*) – The model to use for the batch job.
- **src** (*str*) – The source of the batch job. Currently supports GCS URI(-s) or BigQuery URI. Example: “gs://path/to/input/data” or “bq://projectId.bqDatasetId.bqTableId”.
- **config** (*CreateBatchJobConfig*) – Optional configuration for the batch job.

RETURNS:

A BatchJob object that contains details about the batch job.

Usage:

```
batch_job = await client.aio.batches.create(
    model="gemini-1.5-flash",
    src="gs://path/to/input/data",
)
```

async delete(*, name, config=None)

Skip to content

Deletes a batch job.

RETURN TYPE:

`DeleteResourceJob`

PARAMETERS:

name (*str*) –

A fully-qualified BatchJob resource name or ID. Example: “projects/…/locations/…/batchPredictionJobs/456” or “456”

> when project and location are initialized in the client.

RETURNS:

A DeleteResourceJob object that shows the status of the deletion.

Usage:

```
await client.aio.batches.delete(name='123456789')
```

async get(*, name, config=None)

Gets a batch job.

RETURN TYPE:

`BatchJob`

PARAMETERS:

name (*str*) –

A fully-qualified BatchJob resource name or ID. Example: “projects/…/locations/…/batchPredictionJobs/456” or “456”

> when project and location are initialized in the client.

RETURNS:

A BatchJob object that contains details about the batch job.

Usage:

```
batch_job = await client.aio.batches.get(name='123456789')
print(f"Batch job: {batch_job.name}, state {batch_job.state}")
```

async list(*, config=None)

Skip to content

Lists batch jobs asynchronously.

RETURN TYPE:

`AsyncPager[BatchJob]`

PARAMETERS:

config (*ListBatchJobsConfig*) – Optional configuration for the list request.

RETURNS:

A Pager object that contains one page of batch jobs. When iterating over the pager, it automatically fetches the next page if there are more.

Usage:

```
batch_jobs = await client.aio.batches.list(config={'page_size': 5})
print(f"current page: {batch_jobs.page}")
await batch_jobs_pager.next_page()
print(f"next page: {batch_jobs_pager.page}")
```

class genai.batches.**Batches**(*api_client_*)

Skip to content

Bases: `BaseModule`

`cancel(*, name, config=None)`

Cancels a batch job.

Only available for batch jobs that are running or pending.

RETURN TYPE:

`None`

PARAMETERS:

name (*str*) –

A fully-qualified BatchJob resource name or ID. Example: “projects/…/locations/…/batchPredictionJobs/123456789” or

> ”123456789” when project and location are initialized in the client.

Usage:

```
client.batches.cancel(name='123456789')
```

`create(*, model, src, config=None)`

Creates a batch job.

RETURN TYPE:

`BatchJob`

PARAMETERS:

- **model** (*str*) – The model to use for the batch job.
- **src** (*str*) – The source of the batch job. Currently supports GCS URI(-s) or BigQuery URI. Example: “gs://path/to/input/data” or “bq://projectId.bqDatasetId.bqTableId”.
- **config** (*CreateBatchJobConfig*) – Optional configuration for the batch job.

RETURNS:

A BatchJob object that contains details about the batch job.

Usage:

```
batch_job = client.batches.create(
    model="gemini-1.5-flash",
    src="gs://path/to/input/data",
)
print(batch_job.state)
```

`delete(*, name, config=None)`

Skip to content

Deletes a batch job.

RETURN TYPE:

`DeleteResourceJob`

PARAMETERS:

name (*str*) –

A fully-qualified BatchJob resource name or ID. Example: “projects/…/locations/…/batchPredictionJobs/456” or “456”

> when project and location are initialized in the client.

RETURNS:

A DeleteResourceJob object that shows the status of the deletion.

Usage:

```
client.batches.delete(name='123456789')
```

get(*, name, config=None)

Gets a batch job.

RETURN TYPE:

`BatchJob`

PARAMETERS:

name (*str*) –

A fully-qualified BatchJob resource name or ID. Example: “projects/…/locations/…/batchPredictionJobs/456” or “456”

> when project and location are initialized in the client.

RETURNS:

A BatchJob object that contains details about the batch job.

Usage:

```
batch_job = client.batches.get(name='123456789')
print(f"Batch job: {batch_job.name}, state {batch_job.state}")
```

list(*, config=None)

Skip to content

Lists batch jobs.

RETURN TYPE:

`Pager[BatchJob]`

PARAMETERS:

config (*ListBatchJobsConfig*) – Optional configuration for the list request.

RETURNS:

A Pager object that contains one page of batch jobs. When iterating over the pager, it automatically fetches the next page if there are more.

Usage:

```
batch_jobs = client.batches.list(config={"page_size": 10})
for batch_job in batch_jobs:
  print(f"Batch job: {batch_job.name}, state {batch_job.state}")
```

Skip to content

genai.caches module

class genai.caches.**AsyncCaches**(**api_client_**)

Skip to content

Bases: `BaseModule`

async create(*, model, config=None)

Creates a cached contents resource.

Usage:

```
contents = ... // Initialize the content to cache.
response = await client.aio.caches.create(
    model= ... // The publisher model id
    contents=contents,
    config={
        'display_name': 'test cache',
        'system_instruction': 'What is the sum of the two pdfs?',
        'ttl': '86400s',
    },
)
```

RETURN TYPE:

`CachedContent`

async delete(*, name, config=None)

Deletes cached content.

Usage:

```
await client.aio.caches.delete(name= ... ) // The server-generated
resource name.
```

RETURN TYPE:

`DeleteCachedContentResponse`

async get(*, name, config=None)

Gets cached content configurations.

```
await client.aio.caches.get(name= ... ) // The server-generated resource
name.
```

RETURN TYPE:

`CachedContent`

async list(*, config=None)

RETURN TYPE:

`AsyncPager[CachedContent]`

async update(*, name, config=None)

Skip to content cached content configurations.

```
response = await client.aio.caches.update(
    name= ... // The server-generated resource name.
    config={
        'ttl': '7600s',
    },
)
```

RETURN TYPE:

`CachedContent`

class genai.caches.**Caches**(api_client_)

Skip to content

Bases: `BaseModule`

create(*, model, config=None)

Creates a cached contents resource.

Usage:

```
contents = ... // Initialize the content to cache.
response = client.caches.create(
    model= ... // The publisher model id
    contents=contents,
    config={
        'display_name': 'test cache',
        'system_instruction': 'What is the sum of the two pdfs?',
        'ttl': '86400s',
    },
)
```

RETURN TYPE:

`CachedContent`

delete(*, name, config=None)

Deletes cached content.

Usage:

```
client.caches.delete(name= ... ) // The server-generated resource name.
```

RETURN TYPE:

`DeleteCachedContentResponse`

get(*, name, config=None)

Gets cached content configurations.

```
client.caches.get(name= ... ) // The server-generated resource name.
```

RETURN TYPE:

`CachedContent`

list(*, config=None)

RETURN TYPE:

`Pager[CachedContent]`

update(*, name, config=None)

Updates cached content configurations.

Skip to content

```
response = client.caches.update(
    name= ... // The server-generated resource name.
    config={
        'ttl': '7600s',
    },
)
```

RETURN TYPE:

`CachedContent`

Skip to content

genai.chats module

class genai.chats.**AsyncChat**(*, modules, model, config=None, history)

Bases: `_BaseChat`

Async chat session.

async **send_message**(message, config=None)

Sends the conversation history with the additional message and returns model's response.

RETURN TYPE:

`GenerateContentResponse`

PARAMETERS:

- **message** – The message to send to the model.
- **config** – Optional config to override the default Chat config for this request.

RETURNS:

The model's response.

Usage:

```
chat = client.aio.chats.create(model='gemini-1.5-flash')
response = await chat.send_message('tell me a story')
```

async **send_message_stream**(message, config=None)

Sends the conversation history with the additional message and yields the model's response in chunks.

RETURN TYPE:

`Awaitable[AsyncIterator[GenerateContentResponse]]`

PARAMETERS:

- **message** – The message to send to the model.
- **config** – Optional config to override the default Chat config for this request.

YIELDS:

The model's response in chunks.

Usage:

class genai.chats.**AsyncChats**(modules)

Bases: `object`

Skip to content

A util class to create async chat sessions.

`create(*, model, config=None, history=None)`

Creates a new chat session.

RETURN TYPE:

`AsyncChat`

PARAMETERS:

- **model** – The model to use for the chat.
- **config** – The configuration to use for the generate content request.
- **history** – The history to use for the chat.

RETURNS:

A new chat session.

`class genai.chats.Chat(*, modules, model, config=None, history)`

Skip to content

Bases: `_BaseChat`

Chat session.

send_message(message, config=None)

Sends the conversation history with the additional message and returns the model's response.

RETURN TYPE:

`GenerateContentResponse`

PARAMETERS:

- **message** – The message to send to the model.
- **config** – Optional config to override the default Chat config for this request.

RETURNS:

The model's response.

Usage:

```
chat = client.chats.create(model='gemini-1.5-flash')
response = chat.send_message('tell me a story')
```

send_message_stream(message, config=None)

Sends the conversation history with the additional message and yields the model's response in chunks.

PARAMETERS:

- **message** – The message to send to the model.
- **config** – Optional config to override the default Chat config for this request.

YIELDS:

The model's response in chunks.

Usage:

```
chat = client.chats.create(model='gemini-1.5-flash')
for chunk in chat.send_message_stream('tell me a story'):
  print(chunk.text)
```

class genai.chats.**Chats**(modules)

Bases: `object`

Skip to content

A util class to create chat sessions.

`create(*, model, config=None, history=None)`

Creates a new chat session.

RETURN TYPE:

`Chat`

PARAMETERS:

- **model** – The model to use for the chat.
- **config** – The configuration to use for the generate content request.
- **history** – The history to use for the chat.

RETURNS:

A new chat session.

Skip to content

genai.files module

class genai.files.**AsyncFiles**(api_client_)

Skip to content

Bases: `BaseModule`

async delete(*, name, config=None)

Deletes an existing file from the service.

RETURN TYPE:

`DeleteFileResponse`

PARAMETERS:

- **name** (*str*) – The name identifier for the file to delete.
- **config** (*DeleteFileConfig*) – Optional, configuration for the delete method.

RETURNS:

The response for the delete method

RETURN TYPE:

DeleteFileResponse

Usage:

```
await client.aio.files.delete(name='files/...')
```

async download(*, file, config=None)

Downloads a file's data from the file service.

The Vertex-AI implementation of the API foes not include the file service.

Files created by *upload* can't be downloaded. You can tell which files are downloadable by checking the *download_uri* property.

RETURN TYPE:

`bytes`

PARAMETERS:

- **File** (*str*) – A file name, uri, or file object. Identifying which file to download.
- **config** (*DownloadFileConfigOrDict*) – Optional, configuration for the get method.

RETURNS:

The file data as bytes.

RETURN TYPE:

File

Usage:

Skip to content

```
for file client.files.list():
  if file.download_uri is not None:
    break
else:
  raise ValueError('No files found with a `download_uri`.')
data = client.files.download(file=file)
# data = client.files.download(file=file.name)
# data = client.files.download(file=file.uri)
```

async get(*, name, config=None)

Retrieves the file information from the service.

RETURN TYPE:

`File`

PARAMETERS:

- **name** (*str*) – The name identifier for the file to retrieve.
- **config** (*GetFileConfig*) – Optional, configuration for the get method.

RETURNS:

The file information.

RETURN TYPE:

File

Usage:

```
file = await client.aio.files.get(name='files/...')
print(file.uri)
```

async list(*, config=None)

RETURN TYPE:

`AsyncPager[File]`

async upload(*, file, config=None)

Calls the API to upload a file asynchronously using a supported file service.

RETURN TYPE:

`File`

PARAMETERS:

- **file** – A path to the file or an *IOBase* object to be uploaded. If it's an IOBase object, it must be opened in blocking (the default) mode and binary mode. In other words, do not use non-blocking mode or text mode. The given stream must be seekable, that is, it must be able to call *seek()* on 'path'.
- **config** – Optional parameters to set *diplay_name*, *mime_type*, and *name*.

Skip to content .es.**Files**(api_client_)

Bases: `BaseModule`

delete(*, name, config=None)

Deletes an existing file from the service.

RETURN TYPE:

`DeleteFileResponse`

PARAMETERS:

- **name** (*str*) – The name identifier for the file to delete.
- **config** (*DeleteFileConfig*) – Optional, configuration for the delete method.

RETURNS:

The response for the delete method

RETURN TYPE:

DeleteFileResponse

Usage:

```
client.files.delete(name='files/...')
```

download(*, file, config=None)

Downloads a file's data from storage.

Files created by *upload* can't be downloaded. You can tell which files are downloadable by checking the *source* or *download_uri* property.

Note: This method returns the data as bytes. For *Video* and *GeneratedVideo* objects there is an additional side effect, that it also sets the *video_bytes* property on the *Video* object.

RETURN TYPE:

`bytes`

PARAMETERS:

- **file** (*str*) – A file name, uri, or file object. Identifying which file to download.
- **config** (*DownloadFileConfigOrDict*) – Optional, configuration for the get method.

RETURNS:

The file data as bytes.

RETURN TYPE:

File

Usage:

Skip to content

```
for file client.files.list():
  if file.download_uri is not None:
    break
else:
  raise ValueError('No files found with a `download_uri`.')
data = client.files.download(file=file)
# data = client.files.download(file=file.name)
# data = client.files.download(file=file.download_uri)

video = types.Video(uri=file.uri)
video_bytes = client.files.download(file=video)
video.video_bytes
```

get(*, name, config=None)

Retrieves the file information from the service.

RETURN TYPE:

`File`

PARAMETERS:

- **name** (*str*) – The name identifier for the file to retrieve.
- **config** (*GetFileConfig*) – Optional, configuration for the get method.

RETURNS:

The file information.

RETURN TYPE:

File

Usage:

```
file = client.files.get(name='files/...')
print(file.uri)
```

list(*, config=None)

RETURN TYPE:

`Pager[File]`

upload(*, file, config=None)

Calls the API to upload a file using a supported file service.

RETURN TYPE:

`File`

PARAMETERS:

- **file** – A path to the file or an *IOBase* object to be uploaded. If it's an IOBase object, it must be opened in blocking (the default) mode and binary mode. In other words, do not use non-blocking mode or text mode. The given stream must be seekable, that is, it must be able to call *seek()* on 'path'.
- **config** – Optional parameters to set *diplay_name*, *mime_type*, and *name*.

Skip to content

genai.live module

Live client. The live module is experimental.

class genai.live.**AsyncLive**(api_client_)

Bases: `BaseModule`

AsyncLive. The live module is experimental.

connect(*, model, config=None)

Connect to the live server.

The live module is experimental.

Usage:

```
client = genai.Client(api_key=API_KEY)
config = {}
async with client.aio.live.connect(model='...', config=config) as session:
  await session.send(input='Hello world!', end_of_turn=True)
  async for message in session.receive():
    print(message)
```

RETURN TYPE:

`AsyncIterator`[`AsyncSession`]

class genai.live.**AsyncSession**(api_client, websocket)

Skip to content

Bases: `object`

AsyncSession. The live module is experimental.

async close()

async receive()

Receive model responses from the server.

The method will yield the model responses from the server. The returned responses will represent a complete model turn. When the returned message is function call, user must call *send* with the function response to continue the turn.

The live module is experimental.

RETURN TYPE:

`AsyncIterator[LiveServerMessage]`

YIELDS:

The model responses from the server.

Example usage:

```
client = genai.Client(api_key=API_KEY)

async with client.aio.live.connect(model='...') as session:
  await session.send(input='Hello world!', end_of_turn=True)
  async for message in session.receive():
    print(message)
```

async send(*, input=None, end_of_turn=False)

Send input to the model.

The method will send the input request to the server.

PARAMETERS:

- **input** – The input request to the model.
- **end_of_turn** – Whether the input is the last message in a turn.

Example usage:

```
client = genai.Client(api_key=API_KEY)

async with client.aio.live.connect(model='...') as session:
  await session.send(input='Hello world!', end_of_turn=True)
  async for message in session.receive():
    print(message)
```

async start_stream(*, stream, mime_type)

Skip to content e session from a data stream.

The interaction terminates when the input stream is complete. This method will start two async tasks. One task will be used to send the input stream to the model and the other task will be used to receive the responses from the model.

The live module is experimental.

RETURN TYPE:

`AsyncIterator[LiveServerMessage]`

PARAMETERS:

- **stream** – An iterator that yields the model response.
- **mime_type** – The MIME type of the data in the stream.

YIELDS:

The audio bytes received from the model and server response messages.

Example usage:

```
client = genai.Client(api_key=API_KEY)
config = {'response_modalities': ['AUDIO']}
async def audio_stream():
  stream = read_audio()
  for data in stream:
    yield data
async with client.aio.live.connect(model='...', config=config) as session:
  for audio in session.start_stream(stream = audio_stream(),
  mime_type = 'audio/pcm'):
    play_audio_chunk(audio.data)
```

Skip to content

genai.models module

class genai.models.**AsyncModels**(**api_client_**)

Skip to content

Bases: `BaseModule`

async compute_tokens(*, model, contents, config=None)

Given a list of contents, returns a corresponding TokensInfo containing the

list of tokens and list of token ids.

RETURN TYPE:

`ComputeTokensResponse`

PARAMETERS:

- **model** (*str*) – The model to use.
- **contents** (*list[shared.Content]*) – The content to compute tokens for.

Usage:

```
response = await client.aio.models.compute_tokens(
    model='gemini-2.0-flash',
    contents='What is your name?',
)
print(response)
# tokens_info=[TokensInfo(role='user', token_ids=['1841', ...],
# tokens=[b'What', b' is', b' your', b' name', b'?'])]
```

async count_tokens(*, model, contents, config=None)

Counts the number of tokens in the given content.

Multimodal input is supported for Gemini models.

RETURN TYPE:

`CountTokensResponse`

PARAMETERS:

- **model** (*str*) – The model to use for counting tokens.
- **contents** (*list[types.Content]*) – The content to count tokens for.
- **config** (*CountTokensConfig*) – The configuration for counting tokens.

Usage:

```
response = await client.aio.models.count_tokens(
    model='gemini-2.0-flash',
    contents='What is your name?',
)
print(response)
# total_tokens=5 cached_content_token_count=None
```

async delete(*, model, config=None)

RETURN TYPE:

Skip to content `eteModelResponse`

async edit_image(*, model, prompt, reference_images, config=None)

Edits an image based on a text description and configuration.

RETURN TYPE:

`EditImageResponse`

PARAMETERS:

- **model** (*str*) – The model to use.
- **prompt** (*str*) – A text description of the edit to apply to the image. reference_images (list[Union[RawReferenceImage, MaskReferenceImage, ControlReferenceImage, StyleReferenceImage, SubjectReferenceImage]): The reference images for editing.
- **config** (*EditImageConfig*) – Configuration for editing.

Usage:

```
from google.genai.types import RawReferenceImage, MaskReferenceImage

raw_ref_image = RawReferenceImage(
  reference_id=1,
  reference_image=types.Image.from_file(IMAGE_FILE_PATH),
)

mask_ref_image = MaskReferenceImage(
  reference_id=2,
  config=types.MaskReferenceConfig(
      mask_mode='MASK_MODE_FOREGROUND',
      mask_dilation=0.06,
  ),
)
response = await client.aio.models.edit_image(
  model='imagen-3.0-capability-001',
  prompt='man with dog',
  reference_images=[raw_ref_image, mask_ref_image],
  config=types.EditImageConfig(
      edit_mode= "EDIT_MODE_INPAINT_INSERTION",
      number_of_images= 1,
      include_rai_reason= True,
  )
)
response.generated_images[0].image.show()
# Shows a man with a dog instead of a cat.
```

async **embed_content**(*, **model**, **contents**, **config=None**)

Calculates embeddings for the given contents. Only text is supported.

RETURN TYPE:

`EmbedContentResponse`

PARAMETERS:

- **model** (*str*) – The model to use.
- **ents** (*list[Content]*) – The contents to embed.
- **ig** (*EmbedContentConfig*) – Optional configuration for embeddings.

Usage:

```
embeddings = await client.aio.models.embed_content(
    model= 'text-embedding-004',
    contents=[
        'What is your name?',
        'What is your favorite color?',
    ],
    config={
        'output_dimensionality': 64
    },
)
```

async **generate_content**(*, model, contents, config=None)

Makes an API request to generate content using a model.

Some models support multimodal input and output.

Usage:

```
from google.genai import types
from google import genai

client = genai.Client(
    vertexai=True, project='my-project-id', location='us-central1'
)

response = await client.aio.models.generate_content(
    model='gemini-2.0-flash',
    contents='User input: I like bagels. Answer:',
    config=types.GenerateContentConfig(
        system_instruction=
          [
            'You are a helpful language translator.',
            'Your mission is to translate text in English to French.'
          ]
    ),
)
print(response.text)
# J'aime les bagels.
```

RETURN TYPE:

`GenerateContentResponse`

async **generate_content_stream**(*, model, contents, config=None)

Makes an API request to generate content using a model and yields the model's response in chunks.

Skip to content

For the *model* parameter, supported formats for Vertex AI API include: :rtype: `Awaitable[AsyncIterator[GenerateContentResponse]]`

- The Gemini model ID, for example: ‘gemini-2.0-flash’
- The full resource name starts with ‘projects/’, for example: ‘projects/my-project-id/locations/us-central1/publishers/google/models/gemini-2.0-flash’
- The partial resource name with ‘publishers/’, for example: ‘publishers/google/models/gemini-2.0-flash’ or ‘publishers/meta/models/llama-3.1-405b-instruct-maas’
- / separated publisher and model name, for example: ‘google/gemini-2.0-flash’ or ‘meta/llama-3.1-405b-instruct-maas’

For the *model* parameter, supported formats for Gemini API include: - The Gemini model ID, for example: ‘gemini-2.0-flash’ - The model name starts with ‘models/’, for example:

> ‘models/gemini-2.0-flash’

- For tuned models, the model name starts with ‘tunedModels/’, for example: ‘tunedModels/1234567890123456789’

Some models support multimodal input and output.

Usage:

```
from google.genai import types
from google import genai

client = genai.Client(
    vertexai=True, project='my-project-id', location='us-central1'
)

async for chunk in await client.aio.models.generate_content_stream(
  model='gemini-2.0-flash',
  contents='''What is a good name for a flower shop that specializes in
    selling bouquets of dried flowers?'''
):
  print(chunk.text)
# **Elegant & Classic:**
# * The Dried Bloom
# * Everlasting Florals
# * Timeless Petals

async for chunk in awiat client.aio.models.generate_content_stream(
  model='gemini-2.0-flash',
  contents=[
    types.Part.from_text('What is shown in this image?'),
    types.Part.from_uri('gs://generativeai-downloads/images/scones.jpg',
    'image/jpeg')
  ]
):
  print(chunk.text)
# The image shows a flat lay arrangement of freshly baked blueberry
# scones.
```

async **generate_images**(*, model, prompt, config=None)

Generates images based on a text description and configuration.

RETURN TYPE:

`GenerateImagesResponse`

PARAMETERS:

- **model** (*str*) – The model to use.
- **prompt** (*str*) – A text description of the images to generate.
- **config** (*GenerateImagesConfig*) – Configuration for generation.

Usage:

```
response = await client.aio.models.generate_images(
  model='imagen-3.0-generate-002',
  prompt='Man with a dog',
  config=types.GenerateImagesConfig(
      number_of_images= 1,
      include_rai_reason= True,
  )
response.generated_images[0].image.show()
# Shows a man with a dog.
```

Skip to content

`generate_videos(*, model, prompt=None, image=None, config=None)`

Generates videos based on a text description and configuration.

RETURN TYPE:

`GenerateVideosOperation`

PARAMETERS:

- **model** – The model to use.
- **instances** – A list of prompts, images and videos to generate videos from.
- **config** – Configuration for generation.

Usage:

> ``` operation = client.models.generate_videos(
>
> > model=”veo-2.0-generate-001”, prompt=”A neon hologram of a cat driving at top speed”,
>
> ) while not operation.done:
>
> > time.sleep(10) operation = client.operations.get(operation)
>
> operation.result.generated_videos[0].video.uri ```

`async get(*, model, config=None)`

RETURN TYPE:

`Model`

`async list(*, config=None)`

Makes an API request to list the available models.

If *query_base* is set to True in the config or not set (default), the API will return all available base models. If set to False, it will return all tuned models.

RETURN TYPE:

`AsyncPager[Model]`

PARAMETERS:

config (*ListModelsConfigOrDict*) – Configuration for retrieving models.

Usage:

Skip to content

```
response = await client.aio.models.list(config={'page_size': 5})
print(response.page)
# [Model(name='projects/./locations/./models/123', display_name='my_model'

response = await client.aio.models.list(
    config={'page_size': 5, 'query_base': True}
  )
print(response.page)
# [Model(name='publishers/google/models/gemini-2.0-flash-exp' ...
```

async update(*, model, config=None)

RETURN TYPE:

`Model`

async upscale_image(*, model, image, upscale_factor, config=None)

Makes an API request to upscale a provided image.

RETURN TYPE:

`UpscaleImageResponse`

PARAMETERS:

- **model** (*str*) – The model to use.
- **image** (*Image*) – The input image for upscaling.
- **upscale_factor** (*str*) – The factor to upscale the image (x2 or x4).
- **config** (*UpscaleImageConfig*) – Configuration for upscaling.

Usage:

```
from google.genai.types import Image

IMAGE_FILE_PATH="my-image.png"
response = await client.aio.models.upscale_image(
    model='imagen-3.0-generate-001',
    image=types.Image.from_file(IMAGE_FILE_PATH),
    upscale_factor='x2',
)
response.generated_images[0].image.show()
# Opens my-image.png which is upscaled by a factor of 2.
```

class genai.models.**Models**(api_client_)

Skip to content

Bases: `BaseModule`

compute_tokens(*, model, contents, config=None)

Given a list of contents, returns a corresponding TokensInfo containing the

list of tokens and list of token ids.

This method is not supported by the Gemini Developer API.

RETURN TYPE:

`ComputeTokensResponse`

PARAMETERS:

- **model** (*str*) – The model to use.
- **contents** (*list[shared.Content]*) – The content to compute tokens for.

Usage:

```
response = client.models.compute_tokens(
    model='gemini-2.0-flash',
    contents='What is your name?',
)
print(response)
# tokens_info=[TokensInfo(role='user', token_ids=['1841', ...],
# tokens=[b'What', b' is', b' your', b' name', b'?'])]
```

count_tokens(*, model, contents, config=None)

Counts the number of tokens in the given content.

Multimodal input is supported for Gemini models.

RETURN TYPE:

`CountTokensResponse`

PARAMETERS:

- **model** (*str*) – The model to use for counting tokens.
- **contents** (*list[types.Content]*) – The content to count tokens for.
- **config** (*CountTokensConfig*) – The configuration for counting tokens.

Usage:

```
response = client.models.count_tokens(
    model='gemini-2.0-flash',
    contents='What is your name?',
)
print(response)
# total_tokens=5 cached_content_token_count=None
```

Skip to content

odel, config=None)

YPE:

`DeleteModelResponse`

edit_image(*, model, prompt, reference_images, config=None)

Edits an image based on a text description and configuration.

RETURN TYPE:

`EditImageResponse`

PARAMETERS:

- **model** (*str*) – The model to use.
- **prompt** (*str*) – A text description of the edit to apply to the image. reference_images (list[Union[RawReferenceImage, MaskReferenceImage, ControlReferenceImage, StyleReferenceImage, SubjectReferenceImage]): The reference images for editing.
- **config** (*EditImageConfig*) – Configuration for editing.

Usage:

```
from google.genai.types import RawReferenceImage, MaskReferenceImage

raw_ref_image = RawReferenceImage(
  reference_id=1,
  reference_image=types.Image.from_file(IMAGE_FILE_PATH),
)

mask_ref_image = MaskReferenceImage(
  reference_id=2,
  config=types.MaskReferenceConfig(
      mask_mode='MASK_MODE_FOREGROUND',
      mask_dilation=0.06,
  ),
)
response = client.models.edit_image(
  model='imagen-3.0-capability-001',
  prompt='man with dog',
  reference_images=[raw_ref_image, mask_ref_image],
  config=types.EditImageConfig(
      edit_mode= "EDIT_MODE_INPAINT_INSERTION",
      number_of_images= 1,
      include_rai_reason= True,
  )
)
response.generated_images[0].image.show()
# Shows a man with a dog instead of a cat.
```

embed_content(*, model, contents, config=None)

Calculates embeddings for the given contents. Only text is supported.

RETURN TYPE:

`EmbedContentResponse`

PARAMETERS:

Skip to content

el (*str*) – The model to use.

- **contents** (*list[Content]*) – The contents to embed.
- **config** (*EmbedContentConfig*) – Optional configuration for embeddings.

Usage:

```
embeddings = client.models.embed_content(
    model= 'text-embedding-004',
    contents=[
        'What is your name?',
        'What is your favorite color?',
    ],
    config={
        'output_dimensionality': 64
    },
)
```

generate_content(*, model, contents, config=None)

Makes an API request to generate content using a model.

For the *model* parameter, supported formats for Vertex AI API include: :rtype: `GenerateContentResponse`

- The Gemini model ID, for example: ‘gemini-2.0-flash’
- The full resource name starts with ‘projects/’, for example: ‘projects/my-project-id/locations/us-central1/publishers/google/models/gemini-2.0-flash’
- The partial resource name with ‘publishers/’, for example: ‘publishers/google/models/gemini-2.0-flash’ or ‘publishers/meta/models/llama-3.1-405b-instruct-maas’
- / separated publisher and model name, for example: ‘google/gemini-2.0-flash’ or ‘meta/llama-3.1-405b-instruct-maas’

For the *model* parameter, supported formats for Gemini API include: - The Gemini model ID, for example: ‘gemini-2.0-flash’ - The model name starts with ‘models/’, for example:

> ‘models/gemini-2.0-flash’

- For tuned models, the model name starts with ‘tunedModels/’, for example: ‘tunedModels/1234567890123456789’

Some models support multimodal input and output.

Usage:

Skip to content

```
from google.genai import types
from google import genai

client = genai.Client(
    vertexai=True, project='my-project-id', location='us-central1'
)

response = client.models.generate_content(
  model='gemini-2.0-flash',
  contents='''What is a good name for a flower shop that specializes in
    selling bouquets of dried flowers?'''
)
print(response.text)
# **Elegant & Classic:**
# * The Dried Bloom
# * Everlasting Florals
# * Timeless Petals

response = client.models.generate_content(
  model='gemini-2.0-flash',
  contents=[
    types.Part.from_text('What is shown in this image?'),
    types.Part.from_uri('gs://generativeai-downloads/images/scones.jpg',
    'image/jpeg')
  ]
)
print(response.text)
# The image shows a flat lay arrangement of freshly baked blueberry
# scones.
```

generate_content_stream(*, model, contents, config=None)

Makes an API request to generate content using a model and yields the model's response in chunks.

For the *model* parameter, supported formats for Vertex AI API include: :rtype: `Iterator[GenerateContentResponse]`

- The Gemini model ID, for example: 'gemini-2.0-flash'
- The full resource name starts with 'projects/', for example: 'projects/my-project-id/locations/us-central1/publishers/google/models/gemini-2.0-flash'
- The partial resource name with 'publishers/', for example: 'publishers/google/models/gemini-2.0-flash' or 'publishers/meta/models/llama-3.1-405b-instruct-maas'
- / separated publisher and model name, for example: 'google/gemini-2.0-flash' or 'meta/llama-3.1-405b-instruct-maas'

For the *model* parameter, supported formats for Gemini API include: - The Gemini model ID, for example: 'gemini-2.0-flash' - The model name starts with 'models/', for example:

Skip to content

> 'models/gemini-2.0-flash'

- For tuned models, the model name starts with 'tunedModels/', for example: 'tunedModels/1234567890123456789'

Some models support multimodal input and output.

Usage:

```
from google.genai import types
from google import genai

client = genai.Client(
    vertexai=True, project='my-project-id', location='us-central1'
)

for chunk in client.models.generate_content_stream(
  model='gemini-2.0-flash',
  contents='''What is a good name for a flower shop that specializes in
    selling bouquets of dried flowers?'''
):
  print(chunk.text)
# **Elegant & Classic:**
# * The Dried Bloom
# * Everlasting Florals
# * Timeless Petals

for chunk in client.models.generate_content_stream(
  model='gemini-2.0-flash',
  contents=[
    types.Part.from_text('What is shown in this image?'),
    types.Part.from_uri('gs://generativeai-downloads/images/scones.jpg',
    'image/jpeg')
  ]
):
  print(chunk.text)
# The image shows a flat lay arrangement of freshly baked blueberry
# scones.
```

generate_images(*, model, prompt, config=None)

Generates images based on a text description and configuration.

RETURN TYPE:

`GenerateImagesResponse`

PARAMETERS:

- **model** (*str*) – The model to use.
- **prompt** (*str*) – A text description of the images to generate.
- **config** (*GenerateImagesConfig*) – Configuration for generation.

Skip to content

Usage:

```
response = client.models.generate_images(
  model='imagen-3.0-generate-002',
  prompt='Man with a dog',
  config=types.GenerateImagesConfig(
      number_of_images= 1,
      include_rai_reason= True,
  )
)
response.generated_images[0].image.show()
# Shows a man with a dog.
```

generate_videos(*, model, prompt=None, image=None, config=None)

Generates videos based on a text description and configuration.

RETURN TYPE:

`GenerateVideosOperation`

PARAMETERS:

- **model** – The model to use.
- **instances** – A list of prompts, images and videos to generate videos from.
- **config** – Configuration for generation.

Usage:

``` operation = client.models.generate_videos(

> model=”veo-2.0-generate-001”, prompt=”A neon hologram of a cat driving at top speed”,

) while not operation.done:

> time.sleep(10) operation = client.operations.get(operation)

operation.result.generated_videos[0].video.uri ```

**get**(*, model, config=None)

RETURN TYPE:

`Model`

**list**(*, config=None)

Makes an API request to list the available models.

Skip to content
```

If *query_base* is set to True in the config or not set (default), the API will return all available base models. If set to False, it will return all tuned models.

RETURN TYPE:

`Pager[Model]`

PARAMETERS:

config (*ListModelsConfigOrDict*) – Configuration for retrieving models.

Usage:

```
response=client.models.list(config={'page_size': 5})
print(response.page)
# [Model(name='projects/./locations/./models/123', display_name='my_model'

response=client.models.list(config={'page_size': 5, 'query_base': True})
print(response.page)
# [Model(name='publishers/google/models/gemini-2.0-flash-exp' ...
```

update(*, model, config=None)

RETURN TYPE:

`Model`

upscale_image(*, model, image, upscale_factor, config=None)

Makes an API request to upscale a provided image.

RETURN TYPE:

`UpscaleImageResponse`

PARAMETERS:

- **model** (*str*) – The model to use.
- **image** (*Image*) – The input image for upscaling.
- **upscale_factor** (*str*) – The factor to upscale the image (x2 or x4).
- **config** (*UpscaleImageConfig*) – Configuration for upscaling.

Usage:

```
from google.genai.types import Image

IMAGE_FILE_PATH="my-image.png"
response=client.models.upscale_image(
    model='imagen-3.0-generate-001',
    image=types.Image.from_file(IMAGE_FILE_PATH),
    upscale_factor='x2',
)
response.generated_images[0].image.show()
# Opens my-image.png which is upscaled by a factor of 2.
```

Skip to content

genai.tunings module

class genai.tunings.**AsyncTunings**(api_client_)

Bases: `BaseModule`

async **get**(*, name, config=None)

RETURN TYPE:

`TuningJob`

async **list**(*, config=None)

RETURN TYPE:

`AsyncPager`[`TuningJob`]

tune(*, base_model, training_dataset, config=None)

RETURN TYPE:

`TuningJob`

class genai.tunings.**Tunings**(api_client_)

Bases: `BaseModule`

get(*, name, config=None)

RETURN TYPE:

`TuningJob`

list(*, config=None)

RETURN TYPE:

`Pager`[`TuningJob`]

tune(*, base_model, training_dataset, config=None)

RETURN TYPE:

`TuningJob`

Skip to content

genai.types module

class genai.types.**AdapterSize**(**values*)

Bases: `CaseInSensitiveEnum`

Optional. Adapter size for tuning.

ADAPTER_SIZE_EIGHT = 'ADAPTER_SIZE_EIGHT'

ADAPTER_SIZE_FOUR = 'ADAPTER_SIZE_FOUR'

ADAPTER_SIZE_ONE = 'ADAPTER_SIZE_ONE'

ADAPTER_SIZE_SIXTEEN = 'ADAPTER_SIZE_SIXTEEN'

ADAPTER_SIZE_THIRTY_TWO = 'ADAPTER_SIZE_THIRTY_TWO'

ADAPTER_SIZE_UNSPECIFIED = 'ADAPTER_SIZE_UNSPECIFIED'

pydantic model genai.types.**AutomaticFunctionCallingConfig**

Bases: `BaseModel`

The configuration for automatic function calling.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `disable (bool | None)`
- `ignore_call_history (bool | None)`
- `maximum_remote_calls (int | None)`

field disable: `Optional[bool]` = None

Whether to disable automatic function calling. If not set or set to False, will enable automatic function calling. If set to True, will disable automatic function calling.

field ignore_call_history: `Optional[bool]` = None (alias 'ignoreCallHistory')

If automatic function calling is enabled, whether to ignore call history to the response. If not set, SDK will set ignore_call_history to false, and will append the call history to GenerateContentResponse.automatic_function_calling_history.

field maximum_remote_calls: `Optional[int]` = 10 (alias 'maximumRemoteCalls')

If automatic function calling is enabled, maximum number of remote calls for automatic function calling. This number should be a positive integer. If not set, SDK will set maximum number of remote calls to 10.

Skip to content

class genai.types.**AutomaticFunctionCallingConfigDict**

Bases: `TypedDict`

The configuration for automatic function calling.

disable: `Optional[bool]`

Whether to disable automatic function calling. If not set or set to False, will enable automatic function calling. If set to True, will disable automatic function calling.

ignore_call_history: `Optional[bool]`

If automatic function calling is enabled, whether to ignore call history to the response. If not set, SDK will set ignore_call_history to false, and will append the call history to GenerateContentResponse.automatic_function_calling_history.

maximum_remote_calls: `Optional[int]`

If automatic function calling is enabled, maximum number of remote calls for automatic function calling. This number should be a positive integer. If not set, SDK will set maximum number of remote calls to 10.

pydantic model genai.types.**BatchJob**

Skip to content

Bases: `BaseModel`

Config for batches.create return value.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `create_time (datetime.datetime | None)`
- `dest (genai.types.BatchJobDestination | None)`
- `display_name (str | None)`
- `end_time (datetime.datetime | None)`
- `error (genai.types.JobError | None)`
- `model (str | None)`
- `name (str | None)`
- `src (genai.types.BatchJobSource | None)`
- `start_time (datetime.datetime | None)`
- `state (genai.types.JobState | None)`
- `update_time (datetime.datetime | None)`

field create_time: `Optional[datetime] = None (alias 'createTime')`

Output only. Time when the Job was created.

field dest: `Optional[BatchJobDestination] = None`

Configuration for the output data.

field display_name: `Optional[str] = None (alias 'displayName')`

The user-defined name of this Job.

field end_time: `Optional[datetime] = None (alias 'endTime')`

Output only. Time when the Job entered any of the following states: *JOB_STATE_SUCCEEDED*, *JOB_STATE_FAILED*, *JOB_STATE_CANCELLED*.

field error: `Optional[JobError] = None`

Output only. Only populated when the job's state is JOB_STATE_FAILED or JOB_STATE_CANCELLED.

field model: `Optional[str] = None`

The name of the model that produces the predictions via the BatchJob.

Skip to content

field name: `Optional[str] = None`

Output only. Resource name of the Job.

field src: `Optional[BatchJobSource] = None`

Configuration for the input data.

field start_time: `Optional[datetime] = None (alias 'startTime')`

Output only. Time when the Job for the first time entered the *JOB_STATE_RUNNING* state.

field state: `Optional[JobState] = None`

Output only. The detailed state of the job.

field update_time: `Optional[datetime] = None (alias 'updateTime')`

Output only. Time when the Job was most recently updated.

pydantic model genai.types.**BatchJobDestination**

Bases: `BaseModel`

Config for *des* parameter.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `bigquery_uri (str | None)`
- `format (str | None)`
- `gcs_uri (str | None)`

field bigquery_uri: `Optional[str] = None (alias 'bigqueryUri')`

The BigQuery URI to the output table.

field format: `Optional[str] = None`

Storage format of the output files. Must be one of: 'jsonl', 'bigquery'.

field gcs_uri: `Optional[str] = None (alias 'gcsUri')`

The Google Cloud Storage URI to the output file.

class genai.types.**BatchJobDestinationDict**

Skip to content

Bases: `TypedDict`

Config for *des* parameter.

bigquery_uri: `Optional[str]`

The BigQuery URI to the output table.

format: `Optional[str]`

Storage format of the output files. Must be one of: 'jsonl', 'bigquery'.

gcs_uri: `Optional[str]`

The Google Cloud Storage URI to the output file.

class genai.types.**BatchJobDict**

Skip to content

Bases: `TypedDict`

Config for batches.create return value.

create_time: `Optional[datetime]`

Output only. Time when the Job was created.

dest: `Optional[BatchJobDestinationDict]`

Configuration for the output data.

display_name: `Optional[str]`

The user-defined name of this Job.

end_time: `Optional[datetime]`

JOB_STATE_SUCCEEDED, *JOB_STATE_FAILED*, *JOB_STATE_CANCELLED*.

TYPE:

Output only. Time when the Job entered any of the following states

error: `Optional[JobErrorDict]`

Output only. Only populated when the job's state is JOB_STATE_FAILED or JOB_STATE_CANCELLED.

model: `Optional[str]`

The name of the model that produces the predictions via the BatchJob.

name: `Optional[str]`

Output only. Resource name of the Job.

src: `Optional[BatchJobSourceDict]`

Configuration for the input data.

start_time: `Optional[datetime]`

Output only. Time when the Job for the first time entered the *JOB_STATE_RUNNING* state.

state: `Optional[JobState]`

Output only. The detailed state of the job.

update_time: `Optional[datetime]`

Output only. Time when the Job was most recently updated.

pydantic model genai.types.**BatchJobSource**

Bases: `BaseModel`

Config for *src* parameter.

model by parsing and validating input data from keyword arguments.

Skip to content

ationError][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `bigquery_uri (str | None)`
- `format (str | None)`
- `gcs_uri (list[str] | None)`

field bigquery_uri: `Optional[str] = None (alias 'bigqueryUri')`

The BigQuery URI to input table.

field format: `Optional[str] = None`

Storage format of the input files. Must be one of: 'jsonl', 'bigquery'.

field gcs_uri: `Optional[list[str]] = None (alias 'gcsUri')`

The Google Cloud Storage URIs to input files.

class genai.types.**BatchJobSourceDict**

Bases: `TypedDict`

Config for *src* parameter.

bigquery_uri: `Optional[str]`

The BigQuery URI to input table.

format: `Optional[str]`

Storage format of the input files. Must be one of: 'jsonl', 'bigquery'.

gcs_uri: `Optional[list[str]]`

The Google Cloud Storage URIs to input files.

pydantic model genai.types.**Blob**

Bases: `BaseModel`

Content blob.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

Skip to content

FIELDS:

- `data (bytes | None)`
- `mime_type (str | None)`

field data: `Optional[bytes] = None`

Required. Raw bytes.

field mime_type: `Optional[str] = None (alias 'mimeType')`

Required. The IANA standard MIME type of the source data.

class genai.types.**BlobDict**

Bases: `TypedDict`

Content blob.

data: `Optional[bytes]`

Required. Raw bytes.

mime_type: `Optional[str]`

Required. The IANA standard MIME type of the source data.

class genai.types.**BlockedReason**(*values)

Bases: `CaseInSensitiveEnum`

Output only. Blocked reason.

BLOCKED_REASON_UNSPECIFIED = 'BLOCKED_REASON_UNSPECIFIED'

BLOCKLIST = 'BLOCKLIST'

OTHER = 'OTHER'

PROHIBITED_CONTENT = 'PROHIBITED_CONTENT'

SAFETY = 'SAFETY'

pydantic model genai.types.**CachedContent**

Bases: `BaseModel`

A resource used in LLM queries for users to explicitly specify what to cache.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

Skip to content

FIELDS:

- `create_time (datetime.datetime | None)`
- `display_name (str | None)`
- `expire_time (datetime.datetime | None)`
- `model (str | None)`
- `name (str | None)`
- `update_time (datetime.datetime | None)`
- `usage_metadata (genai.types.CachedContentUsageMetadata | None)`

field create_time: `Optional[datetime] = None (alias 'createTime')`

Creation time of the cache entry.

field display_name: `Optional[str] = None (alias 'displayName')`

The user-generated meaningful display name of the cached content.

field expire_time: `Optional[datetime] = None (alias 'expireTime')`

Expiration time of the cached content.

field model: `Optional[str] = None`

The name of the publisher model to use for cached content.

field name: `Optional[str] = None`

The server-generated resource name of the cached content.

field update_time: `Optional[datetime] = None (alias 'updateTime')`

When the cache entry was last updated in UTC time.

field usage_metadata: `Optional[CachedContentUsageMetadata] = None (alias 'usageMetadata')`

Metadata on the usage of the cached content.

class genai.types.**CachedContentDict**

Skip to content

Bases: `TypedDict`

A resource used in LLM queries for users to explicitly specify what to cache.

create_time: `Optional[datetime]`

Creation time of the cache entry.

display_name: `Optional[str]`

The user-generated meaningful display name of the cached content.

expire_time: `Optional[datetime]`

Expiration time of the cached content.

model: `Optional[str]`

The name of the publisher model to use for cached content.

name: `Optional[str]`

The server-generated resource name of the cached content.

update_time: `Optional[datetime]`

When the cache entry was last updated in UTC time.

usage_metadata: `Optional[CachedContentUsageMetadataDict]`

Metadata on the usage of the cached content.

pydantic model genai.types.**CachedContentUsageMetadata**

Skip to content

Bases: `BaseModel`

Metadata on the usage of the cached content.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `audio_duration_seconds (int | None)`
- `image_count (int | None)`
- `text_count (int | None)`
- `total_token_count (int | None)`
- `video_duration_seconds (int | None)`

field audio_duration_seconds: `Optional[int] = None (alias 'audioDurationSeconds')`

Duration of audio in seconds.

field image_count: `Optional[int] = None (alias 'imageCount')`

Number of images.

field text_count: `Optional[int] = None (alias 'textCount')`

Number of text characters.

field total_token_count: `Optional[int] = None (alias 'totalTokenCount')`

Total number of tokens that the cached content consumes.

field video_duration_seconds: `Optional[int] = None (alias 'videoDurationSeconds')`

Duration of video in seconds.

class genai.types.**CachedContentUsageMetadataDict**

Skip to content

Bases: `TypedDict`

Metadata on the usage of the cached content.

audio_duration_seconds: `Optional[int]`

Duration of audio in seconds.

image_count: `Optional[int]`

Number of images.

text_count: `Optional[int]`

Number of text characters.

total_token_count: `Optional[int]`

Total number of tokens that the cached content consumes.

video_duration_seconds: `Optional[int]`

Duration of video in seconds.

pydantic model genai.types.CancelBatchJobConfig

Bases: `BaseModel`

Optional parameters.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `http_options (genai.types.HttpOptions | None)`

field http_options: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

class genai.types.CancelBatchJobConfigDict

Bases: `TypedDict`

Optional parameters.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

pydantic model genai.types.Candidate

Skip to content `odel`

A response candidate generated from the model.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `avg_logprobs (float | None)`
- `citation_metadata (genai.types.CitationMetadata | None)`
- `content (genai.types.Content | None)`
- `finish_message (str | None)`
- `finish_reason (genai.types.FinishReason | None)`
- `grounding_metadata (genai.types.GroundingMetadata | None)`
- `index (int | None)`
- `logprobs_result (genai.types.LogprobsResult | None)`
- `safety_ratings (list[genai.types.SafetyRating] | None)`
- `token_count (int | None)`

field avg_logprobs: `Optional[float] = None (alias 'avgLogprobs')`

Output only. Average log probability score of the candidate.

field citation_metadata: `Optional[CitationMetadata] = None (alias 'citationMetadata')`

Source attribution of the generated content.

field content: `Optional[Content] = None`

Contains the multi-part content of the response.

field finish_message: `Optional[str] = None (alias 'finishMessage')`

Describes the reason the model stopped generating tokens.

field finish_reason: `Optional[FinishReason] = None (alias 'finishReason')`

Output only. The reason why the model stopped generating tokens. If empty, the model has not stopped generating the tokens.

field grounding_metadata: `Optional[GroundingMetadata] = None (alias 'groundingMetadata')`

Output only. Metadata specifies sources used to ground generated content.

field index: `Optional[int] = None`

Output only. Index of the candidate.

Skip to content **logprobs_result:** `Optional[LogprobsResult] = None (alias 'logprobsResult')`

Output only. Log-likelihood scores for the response tokens and top tokens

field safety_ratings: Optional[list[SafetyRating]] = None (alias 'safetyRatings')

Output only. List of ratings for the safety of a response candidate. There is at most one rating per category.

field token_count: Optional[int] = None (alias 'tokenCount')

Number of tokens for this candidate.

class genai.types.CandidateDict

Bases: TypedDict

A response candidate generated from the model.

avg_logprobs: Optional[float]

Output only. Average log probability score of the candidate.

citation_metadata: Optional[CitationMetadataDict]

Source attribution of the generated content.

content: Optional[ContentDict]

Contains the multi-part content of the response.

finish_message: Optional[str]

Describes the reason the model stopped generating tokens.

finish_reason: Optional[FinishReason]

Output only. The reason why the model stopped generating tokens. If empty, the model has not stopped generating the tokens.

grounding_metadata: Optional[GroundingMetadataDict]

Output only. Metadata specifies sources used to ground generated content.

index: Optional[int]

Output only. Index of the candidate.

logprobs_result: Optional[LogprobsResultDict]

Output only. Log-likelihood scores for the response tokens and top tokens

safety_ratings: Optional[list[SafetyRatingDict]]

Output only. List of ratings for the safety of a response candidate. There is at most one rating per category.

token_count: Optional[int]

Number of tokens for this candidate.

pydantic model genai.types.Citation

Skip to content odel

Source attributions for content.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `end_index (int | None)`
- `license (str | None)`
- `publication_date (genai.types.GoogleTypeDate | None)`
- `start_index (int | None)`
- `title (str | None)`
- `uri (str | None)`

field **end_index**: `Optional[int] = None (alias 'endIndex')`

Output only. End index into the content.

field **license**: `Optional[str] = None`

Output only. License of the attribution.

field **publication_date**: `Optional[GoogleTypeDate] = None (alias 'publicationDate')`

Output only. Publication date of the attribution.

field **start_index**: `Optional[int] = None (alias 'startIndex')`

Output only. Start index into the content.

field **title**: `Optional[str] = None`

Output only. Title of the attribution.

field **uri**: `Optional[str] = None`

Output only. Url reference of the attribution.

class genai.types.**CitationDict**

Skip to content

Bases: `TypedDict`

Source attributions for content.

end_index: `Optional[int]`

Output only. End index into the content.

license: `Optional[str]`

Output only. License of the attribution.

publication_date: `Optional[GoogleTypeDateDict]`

Output only. Publication date of the attribution.

start_index: `Optional[int]`

Output only. Start index into the content.

title: `Optional[str]`

Output only. Title of the attribution.

uri: `Optional[str]`

Output only. Url reference of the attribution.

pydantic model genai.types.**CitationMetadata**

Bases: `BaseModel`

Citation information when the model quotes another source.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `citations (list[genai.types.Citation] | None)`

field citations: `Optional[list[Citation]] = None`

Contains citation information when the model directly quotes, at length, from another source. Can include traditional websites and code repositories.

class genai.types.**CitationMetadataDict**

Bases: `TypedDict`

Citation information when the model quotes another source.

Skip to content

`Optional[list[CitationDict]]`

citation information when the model directly quotes, at length, from another source. Can include traditional websites and code repositories.

pydantic model genai.types.**CodeExecutionResult**

Bases: `BaseModel`

Result of executing the [ExecutableCode].

Always follows a *part* containing the [ExecutableCode].

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `outcome (genai.types.Outcome | None)`
- `output (str | None)`

field outcome: `Optional[Outcome] = None`

Required. Outcome of the code execution.

field output: `Optional[str] = None`

Optional. Contains stdout when code execution is successful, stderr or other description otherwise.

class genai.types.**CodeExecutionResultDict**

Bases: `TypedDict`

Result of executing the [ExecutableCode].

Always follows a *part* containing the [ExecutableCode].

outcome: `Optional[Outcome]`

Required. Outcome of the code execution.

output: `Optional[str]`

Optional. Contains stdout when code execution is successful, stderr or other description otherwise.

pydantic model genai.types.**ComputeTokensConfig**

Bases: `BaseModel`

Optional parameters for computing tokens.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `http_options (genai.types.HttpOptions | None)`

field **http_options**: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

class genai.types.**ComputeTokensConfigDict**

Bases: `TypedDict`

Optional parameters for computing tokens.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

pydantic model genai.types.**ComputeTokensResponse**

Bases: `BaseModel`

Response for computing tokens.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `tokens_info (list[genai.types.TokensInfo] | None)`

field **tokens_info**: `Optional[list[TokensInfo]] = None (alias 'tokensInfo')`

Lists of tokens info from the input. A ComputeTokensRequest could have multiple instances with a prompt in each instance. We also need to return lists of tokens info for the request with multiple instances.

class genai.types.**ComputeTokensResponseDict**

Bases: `TypedDict`

Response for computing tokens.

tokens_info: `Optional[list[TokensInfoDict]]`

Lists of tokens info from the input. A ComputeTokensRequest could have multiple instances with a prompt in each instance. We also need to return lists of tokens info for the request with multiple instances.

Skip to content **genai.types.Content**

Bases: `BaseModel`

Contains the multi-part content of a message.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `parts (list[genai.types.Part] | None)`
- `role (str | None)`

field parts: `Optional[list[Part]] = None`

List of parts that constitute a single message. Each part may have a different IANA MIME type.

field role: `Optional[str] = None`

Optional. The producer of the content. Must be either 'user' or 'model'. Useful to set for multi-turn conversations, otherwise can be left blank or unset. If role is not specified, SDK will determine the role.

class genai.types.**ContentDict**

Bases: `TypedDict`

Contains the multi-part content of a message.

parts: `Optional[list[PartDict]]`

List of parts that constitute a single message. Each part may have a different IANA MIME type.

role: `Optional[str]`

Optional. The producer of the content. Must be either 'user' or 'model'. Useful to set for multi-turn conversations, otherwise can be left blank or unset. If role is not specified, SDK will determine the role.

pydantic model genai.types.**ContentEmbedding**

Bases: `BaseModel`

The embedding generated from an input content.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

Skip to content ly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `statistics (genai.types.ContentEmbeddingStatistics | None)`
- `values (list[float] | None)`

field statistics: `Optional[ContentEmbeddingStatistics] = None`

Vertex API only. Statistics of the input text associated with this embedding.

field values: `Optional[list[float]] = None`

A list of floats representing an embedding.

class genai.types.ContentEmbeddingDict

Bases: `TypedDict`

The embedding generated from an input content.

statistics: `Optional[ContentEmbeddingStatisticsDict]`

Vertex API only. Statistics of the input text associated with this embedding.

pydantic model genai.types.ContentEmbeddingStatistics

Bases: `BaseModel`

Statistics of the input text associated with the result of content embedding.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `token_count (float | None)`
- `truncated (bool | None)`

field token_count: `Optional[float] = None (alias 'tokenCount')`

Vertex API only. Number of tokens of the input text.

field truncated: `Optional[bool] = None`

Vertex API only. If the input text was truncated due to having a length longer than the allowed maximum input.

class genai.types.ContentEmbeddingStatisticsDict

Skip to content

Bases: `TypedDict`

Statistics of the input text associated with the result of content embedding.

token_count: `Optional[float]`

Vertex API only. Number of tokens of the input text.

truncated: `Optional[bool]`

Vertex API only. If the input text was truncated due to having a length longer than the allowed maximum input.

pydantic model genai.types.ControlReferenceConfig

Bases: `BaseModel`

Configuration for a Control reference image.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `control_type (genai.types.ControlReferenceType | None)`
- `enable_control_image_computation (bool | None)`

field control_type: `Optional[ControlReferenceType] = None (alias 'controlType')`

The type of control reference image to use.

field enable_control_image_computation: `Optional[bool] = None (alias 'enableControlImageComputation')`

Defaults to False. When set to True, the control image will be computed by the model based on the control type. When set to False, the control image must be provided by the user.

class genai.types.ControlReferenceConfigDict

Bases: `TypedDict`

Configuration for a Control reference image.

control_type: `Optional[ControlReferenceType]`

The type of control reference image to use.

enable_control_image_computation: `Optional[bool]`

Defaults to False. When set to True, the control image will be computed by the model based on the control type. When set to False, the control image must be provided by the user.

Skip to content

pydantic model genai.types.ControlReferenceImage

Bases: `BaseModel`

A control reference image.

The image of the control reference image is either a control image provided by the user, or a regular image which the backend will use to generate a control image of. In the case of the latter, the enable_control_image_computation field in the config should be set to True.

A control image is an image that represents a sketch image of areas for the model to fill in based on the prompt.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `config (genai.types.ControlReferenceConfig | None)`
- `control_image_config (genai.types.ControlReferenceConfig | None)`
- `reference_id (int | None)`
- `reference_image (genai.types.Image | None)`
- `reference_type (str | None)`

VALIDATORS:

- `_validate_mask_image_config` » `all fields`

field config: `Optional[ControlReferenceConfig] = None`

Re-map config to control_reference_config to send to API.

Configuration for the control reference image.

VALIDATED BY:

- `_validate_mask_image_config`

field control_image_config: `Optional[ControlReferenceConfig] = None (alias 'controlImageConfig')`

VALIDATED BY:

- `_validate_mask_image_config`

field reference_id: `Optional[int] = None (alias 'referenceId')`

The id of the reference image.

VALIDATED BY:

- `_validate_mask_image_config`

field reference_image: `Optional[Image] = None (alias 'referenceImage')`

Skip to content

The reference image for the editing operation.

VALIDATED BY:

- `_validate_mask_image_config`

field **reference_type**: `Optional[str] = None (alias 'referenceType')`

The type of the reference image. Only set by the SDK.

VALIDATED BY:

- `_validate_mask_image_config`

class genai.types.**ControlReferenceImageDict**

Bases: `TypedDict`

A control reference image.

The image of the control reference image is either a control image provided by the user, or a regular image which the backend will use to generate a control image of. In the case of the latter, the enable_control_image_computation field in the config should be set to True.

A control image is an image that represents a sketch image of areas for the model to fill in based on the prompt.

config: `Optional[ControlReferenceConfigDict]`

Configuration for the control reference image.

reference_id: `Optional[int]`

The id of the reference image.

reference_image: `Optional[ImageDict]`

The reference image for the editing operation.

reference_type: `Optional[str]`

The type of the reference image. Only set by the SDK.

class genai.types.**ControlReferenceType**(*values)

Bases: `CaseInSensitiveEnum`

Enum representing the control type of a control reference image.

CONTROL_TYPE_CANNY = 'CONTROL_TYPE_CANNY'

CONTROL_TYPE_DEFAULT = 'CONTROL_TYPE_DEFAULT'

CONTROL_TYPE_FACE_MESH = 'CONTROL_TYPE_FACE_MESH'

CONTROL_TYPE_SCRIBBLE = 'CONTROL_TYPE_SCRIBBLE'

pydantic model genai.types.**CountTokensConfig**

`odel`

Skip to content

e count_tokens method.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `generation_config (genai.types.GenerationConfig | None)`
- `http_options (genai.types.HttpOptions | None)`
- `system_instruction (genai.types.Content | list[genai.types.File | genai.types.Part | PIL.Image.Image | str] | genai.types.File | genai.types.Part | PIL.Image.Image | str | None)`
- `tools (list[genai.types.Tool] | None)`

field generation_config: `Optional[GenerationConfig] = None (alias 'generationConfig')`

Configuration that the model uses to generate the response. Not supported by the Gemini Developer API.

field http_options: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

field system_instruction: `Union[Content, list[Union[File, Part, Image, str]], File, Part, Image, str, None] = None (alias 'systemInstruction')`

Instructions for the model to steer it toward better performance.

field tools: `Optional[list[Tool]] = None`

Code that enables the system to interact with external systems to perform an action outside of the knowledge and scope of the model.

class genai.types.**CountTokensConfigDict**

Skip to content

Bases: `TypedDict`

Config for the count_tokens method.

generation_config: `Optional[GenerationConfigDict]`

Configuration that the model uses to generate the response. Not supported by the Gemini Developer API.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

system_instruction: `Union[Content, list[Union[File, Part, Image, str]], File, Part, Image, str, ContentDict, None]`

Instructions for the model to steer it toward better performance.

tools: `Optional[list[ToolDict]]`

Code that enables the system to interact with external systems to perform an action outside of the knowledge and scope of the model.

pydantic model genai.types.**CountTokensResponse**

Bases: `BaseModel`

Response for counting tokens.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `cached_content_token_count (int | None)`
- `total_tokens (int | None)`

field cached_content_token_count: `Optional[int] = None (alias 'cachedContentTokenCount')`

Number of tokens in the cached part of the prompt (the cached content).

field total_tokens: `Optional[int] = None (alias 'totalTokens')`

Total number of tokens.

class genai.types.**CountTokensResponseDict**

Skip to content

Bases: `TypedDict`

Response for counting tokens.

cached_content_token_count: `Optional[int]`

Number of tokens in the cached part of the prompt (the cached content).

total_tokens: `Optional[int]`

Total number of tokens.

pydantic model genai.types.**CreateBatchJobConfig**

Bases: `BaseModel`

Config for optional parameters.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `dest (str | None)`
- `display_name (str | None)`
- `http_options (genai.types.HttpOptions | None)`

field dest: `Optional[str] = None`

GCS or BigQuery URI prefix for the output predictions. Example: “gs://path/to/output/data” or “bq://projectId.bqDatasetId.bqTableId”.

field display_name: `Optional[str] = None (alias 'displayName')`

The user-defined name of this BatchJob.

field http_options: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

class genai.types.**CreateBatchJobConfigDict**

Skip to content

Bases: `TypedDict`

Config for optional parameters.

dest: `Optional[str]`

GCS or BigQuery URI prefix for the output predictions. Example: “gs://path/to/output/data” or “bq://projectId.bqDatasetId.bqTableId”.

display_name: `Optional[str]`

The user-defined name of this BatchJob.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

pydantic model genai.types.**CreateCachedContentConfig**

Skip to content

Bases: `BaseModel`

Optional configuration for cached content creation.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `contents (list[genai.types.Content | list[genai.types.File | genai.types.Part | PIL.Image.Image | str] | genai.types.File | genai.types.Part | PIL.Image.Image | str] | genai.types.Content | list[genai.types.File | genai.types.Part | PIL.Image.Image | str] | genai.types.File | genai.types.Part | PIL.Image.Image | str | None)`
- `display_name (str | None)`
- `expire_time (datetime.datetime | None)`
- `http_options (genai.types.HttpOptions | None)`
- `system_instruction (genai.types.Content | list[genai.types.File | genai.types.Part | PIL.Image.Image | str] | genai.types.File | genai.types.Part | PIL.Image.Image | str | None)`
- `tool_config (genai.types.ToolConfig | None)`
- `tools (list[genai.types.Tool] | None)`
- `ttl (str | None)`

field contents: `Union[list[Union[Content, list[Union[File, Part, Image, str]], File, Part, Image, str]], Content, list[Union[File, Part, Image, str]], File, Part, Image, str, None] = None`

The content to cache.

field display_name: `Optional[str] = None (alias 'displayName')`

The user-generated meaningful display name of the cached content.

field expire_time: `Optional[datetime] = None (alias 'expireTime')`

Timestamp of when this resource is considered expired.

field http_options: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

field system_instruction: `Union[Content, list[Union[File, Part, Image, str]], File, Part, Image, str, None] = None (alias 'systemInstruction')`

er set system instruction.

Skip to content

field tool_config: `Optional[ToolConfig] = None (alias 'toolConfig')`

Configuration for the tools to use. This config is shared for all tools.

field tools: `Optional[list[Tool]] = None`

A list of *Tools* the model may use to generate the next response.

field ttl: `Optional[str] = None`

The TTL for this resource. The expiration time is computed: now + TTL.

class genai.types.**CreateCachedContentConfigDict**

Bases: `TypedDict`

Optional configuration for cached content creation.

contents: `Union[list[Union[Content, list[Union[File, Part, Image, str]], File, Part, Image, str, ContentDict]], Content, list[Union[File, Part, Image, str]], File, Part, Image, str, ContentDict, None]`

The content to cache.

display_name: `Optional[str]`

The user-generated meaningful display name of the cached content.

expire_time: `Optional[datetime]`

Timestamp of when this resource is considered expired.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

system_instruction: `Union[Content, list[Union[File, Part, Image, str]], File, Part, Image, str, ContentDict, None]`

Developer set system instruction.

tool_config: `Optional[ToolConfigDict]`

Configuration for the tools to use. This config is shared for all tools.

tools: `Optional[list[ToolDict]]`

A list of *Tools* the model may use to generate the next response.

ttl: `Optional[str]`

now + TTL.

TYPE:

The TTL for this resource. The expiration time is computed

pydantic model genai.types.**CreateFileConfig**

Bases: `BaseModel`

Used to override the default configuration.

Skip to content

model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `http_options (genai.types.HttpOptions | None)`

field **http_options**: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

class genai.types.**CreateFileConfigDict**

Bases: `TypedDict`

Used to override the default configuration.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

pydantic model genai.types.**CreateFileResponse**

Bases: `BaseModel`

Response for the create file method.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `http_headers (dict[str, str] | None)`

field **http_headers**: `Optional[dict[str, str]] = None (alias 'httpHeaders')`

Used to retain the HTTP headers in the request

class genai.types.**CreateFileResponseDict**

Bases: `TypedDict`

Response for the create file method.

http_headers: `Optional[dict[str, str]]`

Used to retain the HTTP headers in the request

pydantic model genai.types.**CreateTuningJobConfig**

Bases: `BaseModel`

Skip to content ıe-tuning job creation request - optional fields.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `adapter_size (genai.types.AdapterSize | None)`
- `batch_size (int | None)`
- `description (str | None)`
- `epoch_count (int | None)`
- `http_options (genai.types.HttpOptions | None)`
- `learning_rate (float | None)`
- `learning_rate_multiplier (float | None)`
- `tuned_model_display_name (str | None)`
- `validation_dataset (genai.types.TuningValidationDataset | None)`

field adapter_size: `Optional[AdapterSize] = None (alias 'adapterSize')`

Adapter size for tuning.

field batch_size: `Optional[int] = None (alias 'batchSize')`

The batch size hyperparameter for tuning. If not set, a default of 4 or 16 will be used based on the number of training examples.

field description: `Optional[str] = None`

The description of the TuningJob

field epoch_count: `Optional[int] = None (alias 'epochCount')`

Number of complete passes the model makes over the entire training dataset during training.

field http_options: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

field learning_rate: `Optional[float] = None (alias 'learningRate')`

The learning rate hyperparameter for tuning. If not set, a default of 0.001 or 0.0002 will be calculated based on the number of training examples.

field learning_rate_multiplier: `Optional[float] = None (alias 'learningRateMultiplier')`

Multiplier for adjusting the default learning rate.

field tuned_model_display_name: `Optional[str] = None (alias 'tunedModelDisplayName')`

lay name of the tuned Model. The name can be up to 128 characters long and can
of any UTF-8 characters.

Skip to content

field validation_dataset: `Optional[TuningValidationDataset] = None (alias`

'validationDataset')

Cloud Storage path to file containing training dataset for tuning. The dataset must be formatted as a JSONL file.

class genai.types.**CreateTuningJobConfigDict**

Bases: TypedDict

Supervised fine-tuning job creation request - optional fields.

adapter_size: Optional[**AdapterSize**]

Adapter size for tuning.

batch_size: Optional[int]

The batch size hyperparameter for tuning. If not set, a default of 4 or 16 will be used based on the number of training examples.

description: Optional[str]

The description of the TuningJob

epoch_count: Optional[int]

Number of complete passes the model makes over the entire training dataset during training.

http_options: Optional[**HttpOptionsDict**]

Used to override HTTP request options.

learning_rate: Optional[float]

The learning rate hyperparameter for tuning. If not set, a default of 0.001 or 0.0002 will be calculated based on the number of training examples.

learning_rate_multiplier: Optional[float]

Multiplier for adjusting the default learning rate.

tuned_model_display_name: Optional[str]

The display name of the tuned Model. The name can be up to 128 characters long and can consist of any UTF-8 characters.

validation_dataset: Optional[**TuningValidationDatasetDict**]

Cloud Storage path to file containing training dataset for tuning. The dataset must be formatted as a JSONL file.

pydantic model genai.types.**DatasetDistribution**

Bases: BaseModel

Distribution computed over a tuning dataset.

' model by parsing and validating input data from keyword arguments.

Skip to content

ationError][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `buckets (list[genai.types.DatasetDistributionDistributionBucket] | None)`
- `max (float | None)`
- `mean (float | None)`
- `median (float | None)`
- `min (float | None)`
- `p5 (float | None)`
- `p95 (float | None)`
- `sum (float | None)`

field buckets: `Optional[list[DatasetDistributionDistributionBucket]] = None`

Output only. Defines the histogram bucket.

field max: `Optional[float] = None`

Output only. The maximum of the population values.

field mean: `Optional[float] = None`

Output only. The arithmetic mean of the values in the population.

field median: `Optional[float] = None`

Output only. The median of the values in the population.

field min: `Optional[float] = None`

Output only. The minimum of the population values.

field p5: `Optional[float] = None`

Output only. The 5th percentile of the values in the population.

field p95: `Optional[float] = None`

Output only. The 95th percentile of the values in the population.

field sum: `Optional[float] = None`

Output only. Sum of a given population of values.

class genai.types.**DatasetDistributionDict**

Skip to content

Bases: `TypedDict`

Distribution computed over a tuning dataset.

buckets: `Optional[list[DatasetDistributionDistributionBucketDict]]`

Output only. Defines the histogram bucket.

max: `Optional[float]`

Output only. The maximum of the population values.

mean: `Optional[float]`

Output only. The arithmetic mean of the values in the population.

median: `Optional[float]`

Output only. The median of the values in the population.

min: `Optional[float]`

Output only. The minimum of the population values.

p5: `Optional[float]`

Output only. The 5th percentile of the values in the population.

p95: `Optional[float]`

Output only. The 95th percentile of the values in the population.

sum: `Optional[float]`

Output only. Sum of a given population of values.

pydantic model genai.types.**DatasetDistributionDistributionBucket**

Skip to content

Bases: `BaseModel`

Dataset bucket used to create a histogram for the distribution given a population of values.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `count (int | None)`
- `left (float | None)`
- `right (float | None)`

field count: `Optional[int] = None`

Output only. Number of values in the bucket.

field left: `Optional[float] = None`

Output only. Left bound of the bucket.

field right: `Optional[float] = None`

Output only. Right bound of the bucket.

class genai.types.**DatasetDistributionDistributionBucketDict**

Bases: `TypedDict`

Dataset bucket used to create a histogram for the distribution given a population of values.

count: `Optional[int]`

Output only. Number of values in the bucket.

left: `Optional[float]`

Output only. Left bound of the bucket.

right: `Optional[float]`

Output only. Right bound of the bucket.

pydantic model genai.types.**DatasetStats**

Bases: `BaseModel`

Statistics computed over a tuning dataset.

Create a new model by parsing and validating input data from keyword arguments.

Skip to content

tionError][pydantic_core.ValidationError] if the input data cannot be validated to nodel.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `total_billable_character_count (int | None)`
- `total_tuning_character_count (int | None)`
- `tuning_dataset_example_count (int | None)`
- `tuning_step_count (int | None)`
- `user_dataset_examples (list[genai.types.Content] | None)`
- `user_input_token_distribution (genai.types.DatasetDistribution | None)`
- `user_message_per_example_distribution (genai.types.DatasetDistribution | None)`
- `user_output_token_distribution (genai.types.DatasetDistribution | None)`

field `total_billable_character_count`: `Optional[int] = None (alias 'totalBillableCharacterCount')`

Output only. Number of billable characters in the tuning dataset.

field `total_tuning_character_count`: `Optional[int] = None (alias 'totalTuningCharacterCount')`

Output only. Number of tuning characters in the tuning dataset.

field `tuning_dataset_example_count`: `Optional[int] = None (alias 'tuningDatasetExampleCount')`

Output only. Number of examples in the tuning dataset.

field `tuning_step_count`: `Optional[int] = None (alias 'tuningStepCount')`

Output only. Number of tuning steps for this Tuning Job.

field `user_dataset_examples`: `Optional[list[Content]] = None (alias 'userDatasetExamples')`

Output only. Sample user messages in the training dataset uri.

field `user_input_token_distribution`: `Optional[DatasetDistribution] = None (alias 'userInputTokenDistribution')`

Output only. Dataset distributions for the user input tokens.

field `user_message_per_example_distribution`: `Optional[DatasetDistribution] = None (alias 'userMessagePerExampleDistribution')`

Output only. Dataset distributions for the messages per example.

field `user_output_token_distribution`: `Optional[DatasetDistribution] = None (alias 'userOutputTokenDistribution')`

Output only. Dataset distributions for the user output tokens.

class genai.types.**DatasetStatsDict**

Skip to content `Dict`

Statistics computed over a tuning dataset.

total_billable_character_count: Optional[int]

Output only. Number of billable characters in the tuning dataset.

total_tuning_character_count: Optional[int]

Output only. Number of tuning characters in the tuning dataset.

tuning_dataset_example_count: Optional[int]

Output only. Number of examples in the tuning dataset.

tuning_step_count: Optional[int]

Output only. Number of tuning steps for this Tuning Job.

user_dataset_examples: Optional[list[ContentDict]]

Output only. Sample user messages in the training dataset uri.

user_input_token_distribution: Optional[DatasetDistributionDict]

Output only. Dataset distributions for the user input tokens.

user_message_per_example_distribution: Optional[DatasetDistributionDict]

Output only. Dataset distributions for the messages per example.

user_output_token_distribution: Optional[DatasetDistributionDict]

Output only. Dataset distributions for the user output tokens.

pydantic model genai.types.**DeleteBatchJobConfig**

Bases: `BaseModel`

Optional parameters for models.get method.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `http_options (genai.types.HttpOptions | None)`

field http_options: Optional[HttpOptions] = None (alias 'httpOptions')

Used to override HTTP request options.

class genai.types.**DeleteBatchJobConfigDict**

Skip to content `Dict`

Optional parameters for models.get method.

http_options: Optional[HttpOptionsDict]

Used to override HTTP request options.

pydantic model genai.types.**DeleteCachedContentConfig**

Bases: BaseModel

Optional parameters for caches.delete method.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- http_options (genai.types.HttpOptions | None)

field http_options: Optional[HttpOptions] = None (alias 'httpOptions')

Used to override HTTP request options.

class genai.types.**DeleteCachedContentConfigDict**

Bases: TypedDict

Optional parameters for caches.delete method.

http_options: Optional[HttpOptionsDict]

Used to override HTTP request options.

pydantic model genai.types.**DeleteCachedContentResponse**

Bases: BaseModel

Empty response for caches.delete method.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

class genai.types.**DeleteCachedContentResponseDict**

Bases: TypedDict

Skip to content

ıse for caches.delete method.

pydantic model genai.types.**DeleteFileConfig**

Bases: `BaseModel`

Used to override the default configuration.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `http_options (genai.types.HttpOptions | None)`

field http_options: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

class genai.types.**DeleteFileConfigDict**

Bases: `TypedDict`

Used to override the default configuration.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

pydantic model genai.types.**DeleteFileResponse**

Bases: `BaseModel`

Response for the delete file method.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

class genai.types.**DeleteFileResponseDict**

Bases: `TypedDict`

Response for the delete file method.

pydantic model genai.types.**DeleteModelConfig**

Bases: `BaseModel`

Create a new model by parsing and validating input data from keyword arguments.

Skip to content

ationError][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `http_options (genai.types.HttpOptions | None)`

field **http_options**: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

class genai.types.**DeleteModelConfigDict**

Bases: `TypedDict`

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

pydantic model genai.types.**DeleteModelResponse**

Bases: `BaseModel`

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

class genai.types.**DeleteModelResponseDict**

Bases: `TypedDict`

pydantic model genai.types.**DeleteResourceJob**

Skip to content

Bases: `BaseModel`

The return value of delete operation.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `done (bool | None)`
- `error (genai.types.JobError | None)`
- `name (str | None)`

```
field done: Optional[bool] = None
field error: Optional[JobError] = None
field name: Optional[str] = None
```

```
class genai.types.DeleteResourceJobDict
```

Bases: `TypedDict`

The return value of delete operation.

```
done: Optional[bool]
error: Optional[JobErrorDict]
name: Optional[str]
```

```
class genai.types.DeploymentResourcesType(*values)
```

Bases: `CaseInSensitiveEnum`

```
AUTOMATIC_RESOURCES = 'AUTOMATIC_RESOURCES'
DEDICATED_RESOURCES = 'DEDICATED_RESOURCES'
DEPLOYMENT_RESOURCES_TYPE_UNSPECIFIED = 'DEPLOYMENT_RESOURCES_TYPE_UNSPECIFIED'
SHARED_RESOURCES = 'SHARED_RESOURCES'
```

```
pydantic model genai.types.DistillationDataStats
```

Bases: `BaseModel`

Statistics computed for datasets used for distillation.

Create a new model by parsing and validating input data from keyword arguments.

Skip to content

ationError][pydantic_core.ValidationError] if the input data cannot be validated to nodel.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `training_dataset_stats (genai.types.DatasetStats | None)`

field **training_dataset_stats**: `Optional[DatasetStats] = None (alias 'trainingDatasetStats')`

Output only. Statistics computed for the training dataset.

class genai.types.**DistillationDataStatsDict**

Bases: `TypedDict`

Statistics computed for datasets used for distillation.

training_dataset_stats: `Optional[DatasetStatsDict]`

Output only. Statistics computed for the training dataset.

pydantic model genai.types.**DistillationHyperParameters**

Bases: `BaseModel`

Hyperparameters for Distillation.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `adapter_size (genai.types.AdapterSize | None)`
- `epoch_count (int | None)`
- `learning_rate_multiplier (float | None)`

field **adapter_size**: `Optional[AdapterSize] = None (alias 'adapterSize')`

Optional. Adapter size for distillation.

field **epoch_count**: `Optional[int] = None (alias 'epochCount')`

Optional. Number of complete passes the model makes over the entire training dataset during training.

field **learning_rate_multiplier**: `Optional[float] = None (alias 'learningRateMultiplier')`

Optional. Multiplier for adjusting the default learning rate.

Skip to content

ıes.**DistillationHyperParametersDict**

.Dict

Hyperparameters for Distillation.

adapter_size: Optional [AdapterSize]

Optional. Adapter size for distillation.

epoch_count: Optional [int]

Optional. Number of complete passes the model makes over the entire training dataset during training.

learning_rate_multiplier: Optional [float]

Optional. Multiplier for adjusting the default learning rate.

pydantic model genai.types.**DistillationSpec**

Skip to content

Bases: `BaseModel`

Tuning Spec for Distillation.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `base_teacher_model (str | None)`
- `hyper_parameters (genai.types.DistillationHyperParameters | None)`
- `pipeline_root_directory (str | None)`
- `student_model (str | None)`
- `training_dataset_uri (str | None)`
- `tuned_teacher_model_source (str | None)`
- `validation_dataset_uri (str | None)`

field base_teacher_model: `Optional[str] = None (alias 'baseTeacherModel')`

The base teacher model that is being distilled, e.g., “gemini-1.0-pro-002”.

field hyper_parameters: `Optional[DistillationHyperParameters] = None (alias 'hyperParameters')`

Optional. Hyperparameters for Distillation.

field pipeline_root_directory: `Optional[str] = None (alias 'pipelineRootDirectory')`

Required. A path in a Cloud Storage bucket, which will be treated as the root output directory of the distillation pipeline. It is used by the system to generate the paths of output artifacts.

field student_model: `Optional[str] = None (alias 'studentModel')`

The student model that is being tuned, e.g., “google/gemma-2b-1.1-it”.

field training_dataset_uri: `Optional[str] = None (alias 'trainingDatasetUri')`

Required. Cloud Storage path to file containing training dataset for tuning. The dataset must be formatted as a JSONL file.

field tuned_teacher_model_source: `Optional[str] = None (alias 'tunedTeacherModelSource')`

The resource name of the Tuned teacher model. Format: *projects/{project}/locations/{location}/models/{model}*.

Skip to content

ation_dataset_uri: `Optional[str] = None (alias 'validationDatasetUri')`

.. Cloud Storage path to file containing validation dataset for tuning. The dataset must be formatted as a JSONL file.

class genai.types.**DistillationSpecDict**

Bases: `TypedDict`

Tuning Spec for Distillation.

base_teacher_model: `Optional[str]`

The base teacher model that is being distilled, e.g., “gemini-1.0-pro-002”.

hyper_parameters: `Optional[DistillationHyperParametersDict]`

Optional. Hyperparameters for Distillation.

pipeline_root_directory: `Optional[str]`

Required. A path in a Cloud Storage bucket, which will be treated as the root output directory of the distillation pipeline. It is used by the system to generate the paths of output artifacts.

student_model: `Optional[str]`

The student model that is being tuned, e.g., “google/gemma-2b-1.1-it”.

training_dataset_uri: `Optional[str]`

Required. Cloud Storage path to file containing training dataset for tuning. The dataset must be formatted as a JSONL file.

tuned_teacher_model_source: `Optional[str]`

projects/{project}/locations/{location}/models/{model}.

TYPE:

The resource name of the Tuned teacher model. Format

validation_dataset_uri: `Optional[str]`

Optional. Cloud Storage path to file containing validation dataset for tuning. The dataset must be formatted as a JSONL file.

pydantic model genai.types.**DownloadFileConfig**

Bases: `BaseModel`

Used to override the default configuration.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

`tions (genai.types.HttpOptions | None)`

Skip to content

`field http_options: Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

class genai.types.**DownloadFileConfigDict**

Bases: `TypedDict`

Used to override the default configuration.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

pydantic model genai.types.**DynamicRetrievalConfig**

Bases: `BaseModel`

Describes the options to customize dynamic retrieval.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `dynamic_threshold (float | None)`
- `mode (genai.types.DynamicRetrievalConfigMode | None)`

field dynamic_threshold: `Optional[float] = None (alias 'dynamicThreshold')`

Optional. The threshold to be used in dynamic retrieval. If not set, a system default value is used.

field mode: `Optional[DynamicRetrievalConfigMode] = None`

The mode of the predictor to be used in dynamic retrieval.

class genai.types.**DynamicRetrievalConfigDict**

Bases: `TypedDict`

Describes the options to customize dynamic retrieval.

dynamic_threshold: `Optional[float]`

Optional. The threshold to be used in dynamic retrieval. If not set, a system default value is used.

mode: `Optional[DynamicRetrievalConfigMode]`

The mode of the predictor to be used in dynamic retrieval.

class genai.types.**DynamicRetrievalConfigMode**(*values)

Skip to content

Bases: `CaseInSensitiveEnum`

Config for the dynamic retrieval config mode.

```
MODE_DYNAMIC = 'MODE_DYNAMIC'
```

```
MODE_UNSPECIFIED = 'MODE_UNSPECIFIED'
```

pydantic model genai.types.**EditImageConfig**

Skip to content

Bases: `BaseModel`

Configuration for editing an image.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `aspect_ratio (str | None)`
- `base_steps (int | None)`
- `edit_mode (genai.types.EditMode | None)`
- `guidance_scale (float | None)`
- `http_options (genai.types.HttpOptions | None)`
- `include_rai_reason (bool | None)`
- `include_safety_attributes (bool | None)`
- `language (genai.types.ImagePromptLanguage | None)`
- `negative_prompt (str | None)`
- `number_of_images (int | None)`
- `output_compression_quality (int | None)`
- `output_gcs_uri (str | None)`
- `output_mime_type (str | None)`
- `person_generation (genai.types.PersonGeneration | None)`
- `safety_filter_level (genai.types.SafetyFilterLevel | None)`
- `seed (int | None)`

field aspect_ratio: `Optional[str] = None (alias 'aspectRatio')`

Aspect ratio of the generated images.

field base_steps: `Optional[int] = None (alias 'baseSteps')`

The number of sampling steps. A higher value has better image quality, while a lower value has better latency.

field edit_mode: `Optional[EditMode] = None (alias 'editMode')`

Describes the editing mode for the request.

field guidance_scale: `Optional[float] = None (alias 'guidanceScale')`

Controls how much the model adheres to the text prompt. Large values increase output and alignment, but may compromise image quality.

Skip to content

field http_options: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

field include_rai_reason: Optional[bool] = None (alias 'includeRaiReason')

Whether to include the Responsible AI filter reason if the image is filtered out of the response.

field include_safety_attributes: Optional[bool] = None (alias 'includeSafetyAttributes')

Whether to report the safety scores of each image in the response.

field language: Optional[ImagePromptLanguage] = None

Language of the text in the prompt.

field negative_prompt: Optional[str] = None (alias 'negativePrompt')

Description of what to discourage in the generated images.

field number_of_images: Optional[int] = None (alias 'numberOfImages')

Number of images to generate.

field output_compression_quality: Optional[int] = None (alias 'outputCompressionQuality')

Compression quality of the generated image (for `image/jpeg` only).

field output_gcs_uri: Optional[str] = None (alias 'outputGcsUri')

Cloud Storage URI used to store the generated images.

field output_mime_type: Optional[str] = None (alias 'outputMimeType')

MIME type of the generated image.

field person_generation: Optional[PersonGeneration] = None (alias 'personGeneration')

Allows generation of people by the model.

field safety_filter_level: Optional[SafetyFilterLevel] = None (alias 'safetyFilterLevel')

Filter level for safety filtering.

field seed: Optional[int] = None

Random seed for image generation. This is not available when `add_watermark` is set to true.

class genai.types.EditImageConfigDict

Skip to content

Bases: `TypedDict`

Configuration for editing an image.

aspect_ratio: `Optional[str]`

Aspect ratio of the generated images.

base_steps: `Optional[int]`

The number of sampling steps. A higher value has better image quality, while a lower value has better latency.

edit_mode: `Optional[EditMode]`

Describes the editing mode for the request.

guidance_scale: `Optional[float]`

Controls how much the model adheres to the text prompt. Large values increase output and prompt alignment, but may compromise image quality.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

include_rai_reason: `Optional[bool]`

Whether to include the Responsible AI filter reason if the image is filtered out of the response.

include_safety_attributes: `Optional[bool]`

Whether to report the safety scores of each image in the response.

language: `Optional[ImagePromptLanguage]`

Language of the text in the prompt.

negative_prompt: `Optional[str]`

Description of what to discourage in the generated images.

number_of_images: `Optional[int]`

Number of images to generate.

output_compression_quality: `Optional[int]`

Compression quality of the generated image (for `image/jpeg` only).

output_gcs_uri: `Optional[str]`

Cloud Storage URI used to store the generated images.

output_mime_type: `Optional[str]`

MIME type of the generated image.

Skip to content **ration:** `Optional[PersonGeneration]`

Allows generation of people by the model.

safety_filter_level: `Optional[SafetyFilterLevel]`

Filter level for safety filtering.

seed: `Optional[int]`

Random seed for image generation. This is not available when `add_watermark` is set to true.

pydantic model genai.types.EditImageResponse

Bases: `BaseModel`

Response for the request to edit an image.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `generated_images (list[genai.types.GeneratedImage] | None)`

field generated_images: `Optional[list[GeneratedImage]] = None (alias 'generatedImages')`

Generated images.

class genai.types.EditImageResponseDict

Bases: `TypedDict`

Response for the request to edit an image.

generated_images: `Optional[list[GeneratedImageDict]]`

Generated images.

class genai.types.EditMode(*values)

Bases: `CaseInSensitiveEnum`

Enum representing the Imagen 3 Edit mode.

EDIT_MODE_BGSWAP = 'EDIT_MODE_BGSWAP'

EDIT_MODE_CONTROLLED_EDITING = 'EDIT_MODE_CONTROLLED_EDITING'

EDIT_MODE_DEFAULT = 'EDIT_MODE_DEFAULT'

EDIT_MODE_INPAINT_INSERTION = 'EDIT_MODE_INPAINT_INSERTION'

EDIT_MODE_INPAINT_REMOVAL = 'EDIT_MODE_INPAINT_REMOVAL'

EDIT_MODE_OUTPAINT = 'EDIT_MODE_OUTPAINT'

Skip to content

RODUCT_IMAGE = 'EDIT_MODE_PRODUCT_IMAGE'

EDIT_MODE_STYLE = 'EDIT_MODE_STYLE'

pydantic model genai.types.**EmbedContentConfig**

Bases: `BaseModel`

Optional parameters for the embed_content method.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `auto_truncate (bool | None)`
- `http_options (genai.types.HttpOptions | None)`
- `mime_type (str | None)`
- `output_dimensionality (int | None)`
- `task_type (str | None)`
- `title (str | None)`

field **auto_truncate**: `Optional[bool] = None (alias 'autoTruncate')`

Vertex API only. Whether to silently truncate inputs longer than the max sequence length. If this option is set to false, oversized inputs will lead to an INVALID_ARGUMENT error, similar to other text APIs.

field **http_options**: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

field **mime_type**: `Optional[str] = None (alias 'mimeType')`

Vertex API only. The MIME type of the input.

field **output_dimensionality**: `Optional[int] = None (alias 'outputDimensionality')`

Reduced dimension for the output embedding. If set, excessive values in the output embedding are truncated from the end. Supported by newer models since 2024 only. You cannot set this value if using the earlier model (*models/embedding-001*).

field **task_type**: `Optional[str] = None (alias 'taskType')`

Type of task for which the embedding will be used.

field **title**: `Optional[str] = None`

Title for the text. Only applicable when TaskType is *RETRIEVAL_DOCUMENT*.

class genai.types.**EmbedContentConfigDict**

`TypedDict`

Skip to content

Optional parameters for the embed_content method.

auto_truncate: Optional[bool]

Vertex API only. Whether to silently truncate inputs longer than the max sequence length. If this option is set to false, oversized inputs will lead to an INVALID_ARGUMENT error, similar to other text APIs.

http_options: Optional[HttpOptionsDict]

Used to override HTTP request options.

mime_type: Optional[str]

Vertex API only. The MIME type of the input.

output_dimensionality: Optional[int]

Reduced dimension for the output embedding. If set, excessive values in the output embedding are truncated from the end. Supported by newer models since 2024 only. You cannot set this value if using the earlier model (*models/embedding-001*).

task_type: Optional[str]

Type of task for which the embedding will be used.

title: Optional[str]

Title for the text. Only applicable when TaskType is *RETRIEVAL_DOCUMENT*.

pydantic model genai.types.**EmbedContentMetadata**

Bases: `BaseModel`

Request-level metadata for the Vertex Embed Content API.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `billable_character_count (int | None)`

field billable_character_count: Optional[int] = None (alias 'billableCharacterCount')

Vertex API only. The total number of billable characters included in the request.

class genai.types.**EmbedContentMetadataDict**

Bases: TypedDict

Skip to content

Request-level metadata for the Vertex Embed Content API.

billable_character_count: Optional[int]

Vertex API only. The total number of billable characters included in the request.

pydantic model genai.types.**EmbedContentResponse**

Bases: BaseModel

Response for the embed_content method.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- embeddings (list[genai.types.ContentEmbedding] | None)
- metadata (genai.types.EmbedContentMetadata | None)

field embeddings: Optional[list[**ContentEmbedding**]] = None

The embeddings for each request, in the same order as provided in the batch request.

field metadata: Optional[**EmbedContentMetadata**] = None

Vertex API only. Metadata about the request.

class genai.types.**EmbedContentResponseDict**

Bases: TypedDict

Response for the embed_content method.

embeddings: Optional[list[**ContentEmbeddingDict**]]

The embeddings for each request, in the same order as provided in the batch request.

metadata: Optional[**EmbedContentMetadataDict**]

Vertex API only. Metadata about the request.

pydantic model genai.types.**EncryptionSpec**

Bases: BaseModel

Represents a customer-managed encryption key spec that can be applied to a top-level resource.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to

Skip to content model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `kms_key_name (str | None)`

field kms_key_name: `Optional[str] = None (alias 'kmsKeyName')`

Required. The Cloud KMS resource identifier of the customer managed encryption key used to protect a resource. Has the form: *projects/my-project/locations/my-region/keyRings/my-kr/cryptoKeys/my-key*. The key needs to be in the same region as where the compute resource is created.

class genai.types.**EncryptionSpecDict**

Bases: `TypedDict`

Represents a customer-managed encryption key spec that can be applied to a top-level resource.

kms_key_name: `Optional[str]`

projects/my-project/locations/my-region/keyRings/my-kr/cryptoKeys/my-key. The key needs to be in the same region as where the compute resource is created.

TYPE:

Required. The Cloud KMS resource identifier of the customer managed encryption key used to protect a resource. Has the form

pydantic model genai.types.**Endpoint**

Bases: `BaseModel`

An endpoint where you deploy models.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `deployed_model_id (str | None)`
- `name (str | None)`

field deployed_model_id: `Optional[str] = None (alias 'deployedModelId')`

ID of the model that's deployed to the endpoint.

field name: `Optional[str] = None`

Resource name of the endpoint.

Skip to content ...es.**EndpointDict**

Bases: `TypedDict`

An endpoint where you deploy models.

deployed_model_id: `Optional[str]`

ID of the model that's deployed to the endpoint.

name: `Optional[str]`

Resource name of the endpoint.

pydantic model genai.types.ExecutableCode

Bases: `BaseModel`

Code generated by the model that is meant to be executed, and the result returned to the model.

Generated when using the [FunctionDeclaration] tool and [FunctionCallingConfig] mode is set to [Mode.CODE].

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `code (str | None)`
- `language (genai.types.Language | None)`

field code: `Optional[str] = None`

Required. The code to be executed.

field language: `Optional[Language] = None`

Required. Programming language of the *code*.

class genai.types.ExecutableCodeDict

Bases: `TypedDict`

Code generated by the model that is meant to be executed, and the result returned to the model.

Generated when using the [FunctionDeclaration] tool and [FunctionCallingConfig] mode is set to [Mode.CODE].

code: `Optional[str]`

Required. The code to be executed.

language: `Optional[Language]`

Required. Programming language of the *code*.

Skip to content

genai.types.FetchPredictOperationConfig

Bases: `BaseModel`

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `http_options (genai.types.HttpOptions | None)`

field http_options: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

class genai.types.**FetchPredictOperationConfigDict**

Bases: `TypedDict`

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

pydantic model genai.types.**File**

Bases: `BaseModel`

A file uploaded to the API.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

Skip to content

FIELDS:

- `create_time (datetime.datetime | None)`
- `display_name (str | None)`
- `download_uri (str | None)`
- `error (genai.types.FileStatus | None)`
- `expiration_time (datetime.datetime | None)`
- `mime_type (str | None)`
- `name (str | None)`
- `sha256_hash (str | None)`
- `size_bytes (int | None)`
- `source (genai.types.FileSource | None)`
- `state (genai.types.FileState | None)`
- `update_time (datetime.datetime | None)`
- `uri (str | None)`
- `video_metadata (dict[str, Any] | None)`

field create_time: `Optional[datetime] = None (alias 'createTime')`

Output only. The timestamp of when the *File* was created.

field display_name: `Optional[str] = None (alias 'displayName')`

Optional. The human-readable display name for the *File*. The display name must be no more than 512 characters in length, including spaces. Example: ‘Welcome Image’

field download_uri: `Optional[str] = None (alias 'downloadUri')`

Output only. The URI of the *File*, only set for downloadable (generated) files.

field error: `Optional[FileStatus] = None`

Output only. Error status if File processing failed.

field expiration_time: `Optional[datetime] = None (alias 'expirationTime')`

Output only. The timestamp of when the *File* will be deleted. Only set if the *File* is scheduled to expire.

field mime_type: `Optional[str] = None (alias 'mimeType')`

Output only. MIME type of the file.

field name: `Optional[str] = None`

The *File* resource name. The ID (name excluding the “files/” prefix) can contain up to 40 characters that are lowercase alphanumeric or dashes (-). The ID cannot start or end with a dash. If the name is empty on create, a unique name will be generated. Example: *files/123-456*

Skip to content

6_hash: `Optional[str] = None (alias 'sha256Hash')`

Output only. SHA-256 hash of the uploaded bytes. The hash value is encoded in base64 format.

field size_bytes: `Optional[int] = None (alias 'sizeBytes')`

Output only. Size of the file in bytes.

field source: `Optional[FileSource] = None`

Output only. The source of the *File*.

field state: `Optional[FileState] = None`

Output only. Processing state of the File.

field update_time: `Optional[datetime] = None (alias 'updateTime')`

Output only. The timestamp of when the *File* was last updated.

field uri: `Optional[str] = None`

Output only. The URI of the *File*.

field video_metadata: `Optional[dict[str, Any]] = None (alias 'videoMetadata')`

Output only. Metadata for a video.

pydantic model genai.types.**FileData**

Bases: `BaseModel`

URI based data.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `file_uri (str | None)`
- `mime_type (str | None)`

field file_uri: `Optional[str] = None (alias 'fileUri')`

Required. URI.

field mime_type: `Optional[str] = None (alias 'mimeType')`

Required. The IANA standard MIME type of the source data.

class genai.types.**FileDataDict**

Skip to content

Bases: `TypedDict`

URI based data.

file_uri: `Optional[str]`

Required. URI.

mime_type: `Optional[str]`

Required. The IANA standard MIME type of the source data.

class genai.types.**FileDict**

Skip to content

Bases: `TypedDict`

A file uploaded to the API.

create_time: `Optional[datetime]`

Output only. The timestamp of when the *File* was created.

display_name: `Optional[str]`

'Welcome Image'

TYPE:

Optional. The human-readable display name for the *File*. The display name must be no more than 512 characters in length, including spaces. Example

download_uri: `Optional[str]`

Output only. The URI of the *File*, only set for downloadable (generated) files.

error: `Optional[FileStatusDict]`

Output only. Error status if File processing failed.

expiration_time: `Optional[datetime]`

Output only. The timestamp of when the *File* will be deleted. Only set if the *File* is scheduled to expire.

mime_type: `Optional[str]`

Output only. MIME type of the file.

name: `Optional[str]`

files/123-456

TYPE:

The *File* resource name. The ID (name excluding the "files/" prefix) can contain up to 40 characters that are lowercase alphanumeric or dashes (-). The ID cannot start or end with a dash. If the name is empty on create, a unique name will be generated. Example

sha256_hash: `Optional[str]`

Output only. SHA-256 hash of the uploaded bytes. The hash value is encoded in base64 format.

size_bytes: `Optional[int]`

Output only. Size of the file in bytes.

source: `Optional[FileSource]`

Output only. The source of the *File*.

state: `Optional[FileState]`

Skip to content nly. Processing state of the File.

update_time: `Optional[datetime]`

Output only. The timestamp of when the *File* was last updated.

uri: Optional[str]

Output only. The URI of the *File*.

video_metadata: Optional[dict[str, Any]]

Output only. Metadata for a video.

class genai.types.**FileSource**(*values)

Bases: CaseInSensitiveEnum

Source of the File.

GENERATED = 'GENERATED'

SOURCE_UNSPECIFIED = 'SOURCE_UNSPECIFIED'

UPLOADED = 'UPLOADED'

class genai.types.**FileState**(*values)

Bases: CaseInSensitiveEnum

State for the lifecycle of a File.

ACTIVE = 'ACTIVE'

FAILED = 'FAILED'

PROCESSING = 'PROCESSING'

STATE_UNSPECIFIED = 'STATE_UNSPECIFIED'

pydantic model genai.types.**FileStatus**

Skip to content

Bases: `BaseModel`

Status of a File that uses a common error model.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `code (int | None)`
- `details (list[dict[str, Any]] | None)`
- `message (str | None)`

field code: `Optional[int] = None`

The status code. 0 for OK, 1 for CANCELLED

field details: `Optional[list[dict[str, Any]]] = None`

A list of messages that carry the error details. There is a common set of message types for APIs to use.

field message: `Optional[str] = None`

A list of messages that carry the error details. There is a common set of message types for APIs to use.

class genai.types.**FileStatusDict**

Bases: `TypedDict`

Status of a File that uses a common error model.

code: `Optional[int]`

The status code. 0 for OK, 1 for CANCELLED

details: `Optional[list[dict[str, Any]]]`

A list of messages that carry the error details. There is a common set of message types for APIs to use.

message: `Optional[str]`

A list of messages that carry the error details. There is a common set of message types for APIs to use.

class genai.types.**FinishReason**(*values)

Bases: `CaseInSensitiveEnum`

Skip to content

The reason why the model stopped generating tokens. If empty, the model has not stopped generating the tokens.

```
BLOCKLIST = 'BLOCKLIST'
FINISH_REASON_UNSPECIFIED = 'FINISH_REASON_UNSPECIFIED'
MALFORMED_FUNCTION_CALL = 'MALFORMED_FUNCTION_CALL'
MAX_TOKENS = 'MAX_TOKENS'
OTHER = 'OTHER'
PROHIBITED_CONTENT = 'PROHIBITED_CONTENT'
RECITATION = 'RECITATION'
SAFETY = 'SAFETY'
SPII = 'SPII'
STOP = 'STOP'
```

pydantic model genai.types.**FunctionCall**

Bases: `BaseModel`

A function call.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `args (dict[str, Any] | None)`
- `id (str | None)`
- `name (str | None)`

field args: `Optional[dict[str, Any]] = None`

Optional. Required. The function parameters and values in JSON object format. See [FunctionDeclaration.parameters] for parameter details.

field id: `Optional[str] = None`

The unique id of the function call. If populated, the client to execute the *function_call* and return the response with the matching *id*.

field name: `Optional[str] = None`

Required. The name of the function to call. Matches [FunctionDeclaration.name].

Skip to content

es.FunctionCallDict

`Dict`

A function call.

args: `Optional[dict[str, Any]]`

Optional. Required. The function parameters and values in JSON object format. See [FunctionDeclaration.parameters] for parameter details.

id: `Optional[str]`

The unique id of the function call. If populated, the client to execute the *function_call* and return the response with the matching *id*.

name: `Optional[str]`

Required. The name of the function to call. Matches [FunctionDeclaration.name].

pydantic model genai.types.**FunctionCallingConfig**

Bases: `BaseModel`

Function calling config.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `allowed_function_names (list[str] | None)`
- `mode (genai.types.FunctionCallingConfigMode | None)`

field allowed_function_names: `Optional[list[str]] = None (alias 'allowedFunctionNames')`

Optional. Function names to call. Only set when the Mode is ANY. Function names should match [FunctionDeclaration.name]. With mode set to ANY, model will predict a function call from the set of function names provided.

field mode: `Optional[FunctionCallingConfigMode] = None`

Optional. Function calling mode.

class genai.types.**FunctionCallingConfigDict**

Skip to content

Bases: `TypedDict`

Function calling config.

allowed_function_names: `Optional[list[str]]`

Optional. Function names to call. Only set when the Mode is ANY. Function names should match [FunctionDeclaration.name]. With mode set to ANY, model will predict a function call from the set of function names provided.

mode: `Optional[FunctionCallingConfigMode]`

Optional. Function calling mode.

class genai.types.**FunctionCallingConfigMode**(*values)

Bases: `CaseInSensitiveEnum`

Config for the function calling config mode.

ANY = 'ANY'

AUTO = 'AUTO'

MODE_UNSPECIFIED = 'MODE_UNSPECIFIED'

NONE = 'NONE'

pydantic model genai.types.**FunctionDeclaration**

Skip to content

Bases: `BaseModel`

Defines a function that the model can generate JSON inputs for.

The inputs are based on OpenAPI 3.0 specifications.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `description (str | None)`
- `name (str | None)`
- `parameters (genai.types.Schema | None)`
- `response (genai.types.Schema | None)`

field description: `Optional[str] = None`

Optional. Description and purpose of the function. Model uses it to decide how and whether to call the function.

field name: `Optional[str] = None`

Required. The name of the function to call. Must start with a letter or an underscore. Must be a-z, A-Z, 0-9, or contain underscores, dots and dashes, with a maximum length of 64.

field parameters: `Optional[Schema] = None`

Optional. Describes the parameters to this function in JSON Schema Object format. Reflects the Open API 3.03 Parameter Object. string Key: the name of the parameter. Parameter names are case sensitive. Schema Value: the Schema defining the type used for the parameter. For function with no parameters, this can be left unset. Parameter names must start with a letter or an underscore and must only contain chars a-z, A-Z, 0-9, or underscores with a maximum length of 64. Example with 1 required and 1 optional parameter: type: OBJECT properties: param1: type: STRING param2: type: INTEGER required: - param1

field response: `Optional[Schema] = None`

Describes the output from the function in the OpenAPI JSON Schema Object format.

classmethod from_callable`(*, client, callable)`

Converts a Callable to a FunctionDeclaration based on the client.

RETURN TYPE:

`FunctionDeclaration`

classmethod from_callable_with_api_option`(*, callable, api_option='GEMINI_API')`

Converts a Callable to a FunctionDeclaration based on the API option.

Skip to content

Supported API option is ‘VERTEX_AI’ or ‘GEMINI_API’. If api_option is unset, it will default to ‘GEMINI_API’. If unsupported api_option is provided, it will raise ValueError.

RETURN TYPE:

`FunctionDeclaration`

class genai.types.**FunctionDeclarationDict**

Bases: `TypedDict`

Defines a function that the model can generate JSON inputs for.

The inputs are based on OpenAPI 3.0 specifications.

description: `Optional[str]`

Optional. Description and purpose of the function. Model uses it to decide how and whether to call the function.

name: `Optional[str]`

Required. The name of the function to call. Must start with a letter or an underscore. Must be a-z, A-Z, 0-9, or contain underscores, dots and dashes, with a maximum length of 64.

parameters: `Optional[SchemaDict]`

the Schema defining the type used for the parameter. For function with no parameters, this can be left unset. Parameter names must start with a letter or an underscore and must only contain chars a-z, A-Z, 0-9, or underscores with a maximum length of 64. Example with 1 required and 1 optional parameter: type: OBJECT properties: param1: type: STRING param2: type: INTEGER required: - param1

TYPE:

Optional. Describes the parameters to this function in JSON Schema Object format. Reflects the Open API 3.03 Parameter Object. string Key

TYPE:

the name of the parameter. Parameter names are case sensitive. Schema Value

response: `Optional[SchemaDict]`

Describes the output from the function in the OpenAPI JSON Schema Object format.

pydantic model genai.types.**FunctionResponse**

Bases: `BaseModel`

A function response.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

Skip to content ly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `id (str | None)`
- `name (str | None)`
- `response (dict[str, Any] | None)`

field id: `Optional[str] = None`

The id of the function call this response is for. Populated by the client to match the corresponding function call *id*.

field name: `Optional[str] = None`

Required. The name of the function to call. Matches [FunctionDeclaration.name] and [FunctionCall.name].

field response: `Optional[dict[str, Any]] = None`

Required. The function response in JSON object format. Use “output” key to specify function output and “error” key to specify error details (if any). If “output” and “error” keys are not specified, then whole “response” is treated as function output.

class genai.types.**FunctionResponseDict**

Bases: `TypedDict`

A function response.

id: `Optional[str]`

The id of the function call this response is for. Populated by the client to match the corresponding function call *id*.

name: `Optional[str]`

Required. The name of the function to call. Matches [FunctionDeclaration.name] and [FunctionCall.name].

response: `Optional[dict[str, Any]]`

Required. The function response in JSON object format. Use “output” key to specify function output and “error” key to specify error details (if any). If “output” and “error” keys are not specified, then whole “response” is treated as function output.

pydantic model genai.types.**GenerateContentConfig**

Bases: `BaseModel`

Optional model configuration parameters.

For more information, see Content generation parameters.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

Skip to content

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `audio_timestamp (bool | None)`
- `automatic_function_calling (genai.types.AutomaticFunctionCallingConfig | None)`
- `cached_content (str | None)`
- `candidate_count (int | None)`
- `frequency_penalty (float | None)`
- `http_options (genai.types.HttpOptions | None)`
- `labels (dict[str, str] | None)`
- `logprobs (int | None)`
- `max_output_tokens (int | None)`
- `media_resolution (genai.types.MediaResolution | None)`
- `presence_penalty (float | None)`
- `response_logprobs (bool | None)`
- `response_mime_type (str | None)`
- `response_modalities (list[str] | None)`
- `response_schema (dict | type | genai.types.Schema | types.GenericAlias | types.UnionType | _UnionGenericAlias | None)`
- `routing_config (genai.types.GenerationConfigRoutingConfig | None)`
- `safety_settings (list[genai.types.SafetySetting] | None)`
- `seed (int | None)`
- `speech_config (genai.types.SpeechConfig | str | None)`
- `stop_sequences (list[str] | None)`
- `system_instruction (genai.types.Content | list[genai.types.File | genai.types.Part | PIL.Image.Image | str] | genai.types.File | genai.types.Part | PIL.Image.Image | str | None)`
- `temperature (float | None)`
- `thinking_config (genai.types.ThinkingConfig | None)`
- `tool_config (genai.types.ToolConfig | None)`
- `tools (list[genai.types.Tool | Callable] | None)`
- `top_k (float | None)`
- `top_p (float | None)`

VALIDATORS:

- `_convert_literal_to_enum` » `response_schema`

field audio_timestamp: `Optional[bool] = None (alias 'audioTimestamp')`

If enabled, audio timestamp will be included in the request to the model.

Skip to content

atic_function_calling: `Optional[AutomaticFunctionCallingConfig] = None (alias 'automaticFunctionCalling')`

The configuration for automatic function calling.

field cached_content: Optional[str] = None (alias 'cachedContent')

Resource name of a context cache that can be used in subsequent requests.

field candidate_count: Optional[int] = None (alias 'candidateCount')

Number of response variations to return.

field frequency_penalty: Optional[float] = None (alias 'frequencyPenalty')

Positive values penalize tokens that repeatedly appear in the generated text, increasing the probability of generating more diverse content.

field http_options: Optional[HttpOptions] = None (alias 'httpOptions')

Used to override HTTP request options.

field labels: Optional[dict[str, str]] = None

Labels with user-defined metadata to break down billed charges.

field logprobs: Optional[int] = None

Number of top candidate tokens to return the log probabilities for at each generation step.

field max_output_tokens: Optional[int] = None (alias 'maxOutputTokens')

Maximum number of tokens that can be generated in the response.

field media_resolution: Optional[MediaResolution] = None (alias 'mediaResolution')

If specified, the media resolution specified will be used.

field presence_penalty: Optional[float] = None (alias 'presencePenalty')

Positive values penalize tokens that already appear in the generated text, increasing the probability of generating more diverse content.

field response_logprobs: Optional[bool] = None (alias 'responseLogprobs')

Whether to return the log probabilities of the tokens that were chosen by the model at each step.

field response_mime_type: Optional[str] = None (alias 'responseMimeType')

Output response media type of the generated candidate text.

field response_modalities: Optional[list[str]] = None (alias 'responseModalities')

The requested modalities of the response. Represents the set of modalities that the model can return.

field response_schema: Union[dict, type, Schema, GenericAlias, , _UnionGenericAlias, None] = None (alias 'responseSchema')

Schema that the generated candidate text must adhere to.

VALIDATED BY:

Skip to content ıvert_literal_to_enum

field routing_config: Optional[GenerationConfigRoutingConfig] = None (alias 'routingConfig')

Configuration for model router requests.

field safety_settings: `Optional[list[SafetySetting]] = None (alias 'safetySettings')`

Safety settings in the request to block unsafe content in the response.

field seed: `Optional[int] = None`

When `seed` is fixed to a specific number, the model makes a best effort to provide the same response for repeated requests. By default, a random number is used.

field speech_config: `Union[SpeechConfig, str, None] = None (alias 'speechConfig')`

The speech generation configuration.

field stop_sequences: `Optional[list[str]] = None (alias 'stopSequences')`

List of strings that tells the model to stop generating text if one of the strings is encountered in the response.

field system_instruction: `Union[Content, list[Union[File, Part, Image, str]], File, Part, Image, str, None] = None (alias 'systemInstruction')`

Instructions for the model to steer it toward better performance. For example, “Answer as concisely as possible” or “Don’t use technical terms in your response”.

field temperature: `Optional[float] = None`

Value that controls the degree of randomness in token selection. Lower temperatures are good for prompts that require a less open-ended or creative response, while higher temperatures can lead to more diverse or creative results.

field thinking_config: `Optional[ThinkingConfig] = None (alias 'thinkingConfig')`

The thinking features configuration.

field tool_config: `Optional[ToolConfig] = None (alias 'toolConfig')`

Associates model output to a specific function call.

field tools: `Optional[list[Union[Tool, Callable]]] = None`

Code that enables the system to interact with external systems to perform an action outside of the knowledge and scope of the model.

field top_k: `Optional[float] = None (alias 'topK')`

For each token selection step, the `top_k` tokens with the highest probabilities are sampled. Then tokens are further filtered based on `top_p` with the final token selected using temperature sampling. Use a lower number for less random responses and a higher number for more random responses.

field top_p: `Optional[float] = None (alias 'topP')`

Tokens are selected from the most to least probable until the sum of their probabilities equals this value. Use a lower value for less random responses and a higher value for more

Skip to content esponses.

class genai.types.**GenerateContentConfigDict**

Bases: `TypedDict`

Optional model configuration parameters.

For more information, see Content generation parameters.

audio_timestamp: `Optional[bool]`

If enabled, audio timestamp will be included in the request to the model.

automatic_function_calling: `Optional[AutomaticFunctionCallingConfigDict]`

The configuration for automatic function calling.

cached_content: `Optional[str]`

Resource name of a context cache that can be used in subsequent requests.

candidate_count: `Optional[int]`

Number of response variations to return.

frequency_penalty: `Optional[float]`

Positive values penalize tokens that repeatedly appear in the generated text, increasing the probability of generating more diverse content.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

labels: `Optional[dict[str, str]]`

Labels with user-defined metadata to break down billed charges.

logprobs: `Optional[int]`

Number of top candidate tokens to return the log probabilities for at each generation step.

max_output_tokens: `Optional[int]`

Maximum number of tokens that can be generated in the response.

media_resolution: `Optional[MediaResolution]`

If specified, the media resolution specified will be used.

presence_penalty: `Optional[float]`

Positive values penalize tokens that already appear in the generated text, increasing the probability of generating more diverse content.

response_logprobs: `Optional[bool]`

Whether to return the log probabilities of the tokens that were chosen by the model at each step.

response_mime_type: `Optional[str]`

Skip to content esponse media type of the generated candidate text.

response_modalities: `Optional[list[str]]`

The requested modalities of the response. Represents the set of modalities that the model can return.

response_schema: Union[dict, type, Schema, GenericAlias, , _UnionGenericAlias, SchemaDict, None]

Schema that the generated candidate text must adhere to.

routing_config: Optional[GenerationConfigRoutingConfigDict]

Configuration for model router requests.

safety_settings: Optional[list[SafetySettingDict]]

Safety settings in the request to block unsafe content in the response.

seed: Optional[int]

When `seed` is fixed to a specific number, the model makes a best effort to provide the same response for repeated requests. By default, a random number is used.

speech_config: Union[SpeechConfig, str, SpeechConfigDict, None]

The speech generation configuration.

stop_sequences: Optional[list[str]]

List of strings that tells the model to stop generating text if one of the strings is encountered in the response.

system_instruction: Union[Content, list[Union[File, Part, Image, str]], File, Part, Image, str, ContentDict, None]

Instructions for the model to steer it toward better performance. For example, “Answer as concisely as possible” or “Don’t use technical terms in your response”.

temperature: Optional[float]

Value that controls the degree of randomness in token selection. Lower temperatures are good for prompts that require a less open-ended or creative response, while higher temperatures can lead to more diverse or creative results.

thinking_config: Optional[ThinkingConfigDict]

The thinking features configuration.

tool_config: Optional[ToolConfigDict]

Associates model output to a specific function call.

tools: Optional[list[Union[ToolDict, Callable]]]

Code that enables the system to interact with external systems to perform an action outside of the knowledge and scope of the model.

top_k: Optional[float]

token selection step, the `top_k` tokens with the highest probabilities are sampled. ens are further filtered based on `top_p` with the final token selected using temperature sampling. Use a lower number for less random responses and a higher number for more random responses.

top_p: `Optional[float]`

Tokens are selected from the most to least probable until the sum of their probabilities equals this value. Use a lower value for less random responses and a higher value for more random responses.

pydantic model genai.types.**GenerateContentResponse**

Skip to content

Bases: `BaseModel`

Response message for PredictionService.GenerateContent.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `automatic_function_calling_history (list[genai.types.Content] | None)`
- `candidates (list[genai.types.Candidate] | None)`
- `create_time (datetime.datetime | None)`
- `model_version (str | None)`
- `parsed (pydantic.main.BaseModel | dict | enum.Enum | None)`
- `prompt_feedback (genai.types.GenerateContentResponsePromptFeedback | None)`
- `response_id (str | None)`
- `usage_metadata (genai.types.GenerateContentResponseUsageMetadata | None)`

field automatic_function_calling_history: `Optional[list[Content]] = None (alias 'automaticFunctionCallingHistory')`

field candidates: `Optional[list[Candidate]] = None`

Response variations returned by the model.

field create_time: `Optional[datetime] = None (alias 'createTime')`

Timestamp when the request is made to the server.

field model_version: `Optional[str] = None (alias 'modelVersion')`

Output only. The model version used to generate the response.

field parsed: `Union[BaseModel, dict, Enum, None] = None`

Parsed response if response_schema is provided. Not available for streaming.

field prompt_feedback: `Optional[GenerateContentResponsePromptFeedback] = None (alias 'promptFeedback')`

Output only. Content filter results for a prompt sent in the request. Note: Sent only in the first stream chunk. Only happens when no candidates were generated due to content violations.

field response_id: `Optional[str] = None (alias 'responseId')`

Identifier for each response.

field usage_metadata: `Optional[GenerateContentResponseUsageMetadata] = None (alias 'usageMetadata')`

Usage metadata about the response(s).

property **code_execution_result**: str | None

Returns the code execution result in the response.

property **executable_code**: str | None

Returns the executable code in the response.

property **function_calls**: list[FunctionCall] | None

Returns the list of function calls in the response.

property **text**: str | None

Returns the concatenation of all text parts in the response.

class genai.types.**GenerateContentResponseDict**

Bases: TypedDict

Response message for PredictionService.GenerateContent.

candidates: Optional[list[CandidateDict]]

Response variations returned by the model.

create_time: Optional[datetime]

Timestamp when the request is made to the server.

model_version: Optional[str]

Output only. The model version used to generate the response.

prompt_feedback: Optional[GenerateContentResponsePromptFeedbackDict]

Sent only in the first stream chunk. Only happens when no candidates were generated due to content violations.

TYPE:

Output only. Content filter results for a prompt sent in the request. Note

response_id: Optional[str]

Identifier for each response.

usage_metadata: Optional[GenerateContentResponseUsageMetadataDict]

Usage metadata about the response(s).

pydantic model genai.types.**GenerateContentResponsePromptFeedback**

Bases: BaseModel

Content filter results for a prompt sent in the request.

Create a new model by parsing and validating input data from keyword arguments.

Skip to content

ɐtionError][pydantic_core.ValidationError] if the input data cannot be validated to nodel.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `block_reason (genai.types.BlockedReason | None)`
- `block_reason_message (str | None)`
- `safety_ratings (list[genai.types.SafetyRating] | None)`

field block_reason: `Optional[BlockedReason] = None (alias 'blockReason')`

Output only. Blocked reason.

field block_reason_message: `Optional[str] = None (alias 'blockReasonMessage')`

Output only. A readable block reason message.

field safety_ratings: `Optional[list[SafetyRating]] = None (alias 'safetyRatings')`

Output only. Safety ratings.

class genai.types.**GenerateContentResponsePromptFeedbackDict**

Bases: `TypedDict`

Content filter results for a prompt sent in the request.

block_reason: `Optional[BlockedReason]`

Output only. Blocked reason.

block_reason_message: `Optional[str]`

Output only. A readable block reason message.

safety_ratings: `Optional[list[SafetyRatingDict]]`

Output only. Safety ratings.

pydantic model genai.types.**GenerateContentResponseUsageMetadata**

Bases: `BaseModel`

Usage metadata about response(s).

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

Skip to content

FIELDS:

- `cached_content_token_count (int | None)`
- `candidates_token_count (int | None)`
- `prompt_token_count (int | None)`
- `total_token_count (int | None)`

field **cached_content_token_count**: `Optional[int] = None (alias 'cachedContentTokenCount')`

Output only. Number of tokens in the cached part in the input (the cached content).

field **candidates_token_count**: `Optional[int] = None (alias 'candidatesTokenCount')`

Number of tokens in the response(s).

field **prompt_token_count**: `Optional[int] = None (alias 'promptTokenCount')`

Number of tokens in the request. When *cached_content* is set, this is still the total effective prompt size meaning this includes the number of tokens in the cached content.

field **total_token_count**: `Optional[int] = None (alias 'totalTokenCount')`

Total token count for prompt and response candidates.

class genai.types.**GenerateContentResponseUsageMetadataDict**

Bases: `TypedDict`

Usage metadata about response(s).

cached_content_token_count: `Optional[int]`

Output only. Number of tokens in the cached part in the input (the cached content).

candidates_token_count: `Optional[int]`

Number of tokens in the response(s).

prompt_token_count: `Optional[int]`

Number of tokens in the request. When *cached_content* is set, this is still the total effective prompt size meaning this includes the number of tokens in the cached content.

total_token_count: `Optional[int]`

Total token count for prompt and response candidates.

pydantic model genai.types.**GenerateImagesConfig**

Bases: `BaseModel`

The config for generating an images.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `add_watermark (bool | None)`
- `aspect_ratio (str | None)`
- `enhance_prompt (bool | None)`
- `guidance_scale (float | None)`
- `http_options (genai.types.HttpOptions | None)`
- `include_rai_reason (bool | None)`
- `include_safety_attributes (bool | None)`
- `language (genai.types.ImagePromptLanguage | None)`
- `negative_prompt (str | None)`
- `number_of_images (int | None)`
- `output_compression_quality (int | None)`
- `output_gcs_uri (str | None)`
- `output_mime_type (str | None)`
- `person_generation (genai.types.PersonGeneration | None)`
- `safety_filter_level (genai.types.SafetyFilterLevel | None)`
- `seed (int | None)`

field add_watermark: `Optional[bool] = None (alias 'addWatermark')`

Whether to add a watermark to the generated images.

field aspect_ratio: `Optional[str] = None (alias 'aspectRatio')`

Aspect ratio of the generated images.

field enhance_prompt: `Optional[bool] = None (alias 'enhancePrompt')`

Whether to use the prompt rewriting logic.

field guidance_scale: `Optional[float] = None (alias 'guidanceScale')`

Controls how much the model adheres to the text prompt. Large values increase output and prompt alignment, but may compromise image quality.

field http_options: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

field include_rai_reason: `Optional[bool] = None (alias 'includeRaiReason')`

Whether to include the Responsible AI filter reason if the image is filtered out of the response.

field include_safety_attributes: `Optional[bool] = None (alias 'includeSafetyAttributes')`

Skip to content to report the safety scores of each image in the response.

field language: `Optional[ImagePromptLanguage] = None`

Language of the text in the prompt.

field **negative_prompt**: Optional[str] = None (alias 'negativePrompt')

Description of what to discourage in the generated images.

field **number_of_images**: Optional[int] = None (alias 'numberOfImages')

Number of images to generate.

field **output_compression_quality**: Optional[int] = None (alias 'outputCompressionQuality')

Compression quality of the generated image (for `image/jpeg` only).

field **output_gcs_uri**: Optional[str] = None (alias 'outputGcsUri')

Cloud Storage URI used to store the generated images.

field **output_mime_type**: Optional[str] = None (alias 'outputMimeType')

MIME type of the generated image.

field **person_generation**: Optional[PersonGeneration] = None (alias 'personGeneration')

Allows generation of people by the model.

field **safety_filter_level**: Optional[SafetyFilterLevel] = None (alias 'safetyFilterLevel')

Filter level for safety filtering.

field **seed**: Optional[int] = None

Random seed for image generation. This is not available when `add_watermark` is set to true.

class genai.types.**GenerateImagesConfigDict**

Skip to content

Bases: `TypedDict`

The config for generating an images.

add_watermark: `Optional[bool]`

Whether to add a watermark to the generated images.

aspect_ratio: `Optional[str]`

Aspect ratio of the generated images.

enhance_prompt: `Optional[bool]`

Whether to use the prompt rewriting logic.

guidance_scale: `Optional[float]`

Controls how much the model adheres to the text prompt. Large values increase output and prompt alignment, but may compromise image quality.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

include_rai_reason: `Optional[bool]`

Whether to include the Responsible AI filter reason if the image is filtered out of the response.

include_safety_attributes: `Optional[bool]`

Whether to report the safety scores of each image in the response.

language: `Optional[ImagePromptLanguage]`

Language of the text in the prompt.

negative_prompt: `Optional[str]`

Description of what to discourage in the generated images.

number_of_images: `Optional[int]`

Number of images to generate.

output_compression_quality: `Optional[int]`

Compression quality of the generated image (for `image/jpeg` only).

output_gcs_uri: `Optional[str]`

Cloud Storage URI used to store the generated images.

output_mime_type: `Optional[str]`

MIME type of the generated image.

person_generation: `Optional[PersonGeneration]`

Skip to content eneration of people by the model.

safety_filter_level: `Optional[SafetyFilterLevel]`

Filter level for safety filtering.

seed: `Optional[int]`

Random seed for image generation. This is not available when `add_watermark` is set to true.

pydantic model genai.types.GenerateImagesResponse

Bases: `BaseModel`

The output images response.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `generated_images (list[genai.types.GeneratedImage] | None)`

field generated_images: `Optional[list[GeneratedImage]] = None (alias 'generatedImages')`

List of generated images.

class genai.types.GenerateImagesResponseDict

Bases: `TypedDict`

The output images response.

generated_images: `Optional[list[GeneratedImageDict]]`

List of generated images.

pydantic model genai.types.GenerateVideosConfig

Bases: `BaseModel`

Configuration for generating videos.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

Skip to content

FIELDS:

- `aspect_ratio (str | None)`
- `duration_seconds (int | None)`
- `enhance_prompt (bool | None)`
- `fps (int | None)`
- `http_options (genai.types.HttpOptions | None)`
- `negative_prompt (str | None)`
- `number_of_videos (int | None)`
- `output_gcs_uri (str | None)`
- `person_generation (str | None)`
- `pubsub_topic (str | None)`
- `resolution (str | None)`
- `seed (int | None)`

field aspect_ratio: `Optional[str] = None (alias 'aspectRatio')`

The aspect ratio for the generated video. 16:9 (landscape) and 9:16 (portrait) are supported.

field duration_seconds: `Optional[int] = None (alias 'durationSeconds')`

Duration of the clip for video generation in seconds.

field enhance_prompt: `Optional[bool] = None (alias 'enhancePrompt')`

Whether to use the prompt rewriting logic.

field fps: `Optional[int] = None`

Frames per second for video generation.

field http_options: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

field negative_prompt: `Optional[str] = None (alias 'negativePrompt')`

Optional field in addition to the text content. Negative prompts can be explicitly stated here to help generate the video.

field number_of_videos: `Optional[int] = None (alias 'numberOfVideos')`

Number of output videos.

field output_gcs_uri: `Optional[str] = None (alias 'outputGcsUri')`

The gcs bucket where to save the generated videos.

field person_generation: `Optional[str] = None (alias 'personGeneration')`

Whether allow to generate person videos, and restrict to specific ages. Supported values are: dont_allow, allow_adult.

Skip to content

b_topic: `Optional[str] = None (alias 'pubsubTopic')`

The pubsub topic where to publish the video generation progress.

field **resolution**: `Optional[str] = None`

The resolution for the generated video. 1280x720, 1920x1080 are supported.

field **seed**: `Optional[int] = None`

The RNG seed. If RNG seed is exactly same for each request with unchanged inputs, the prediction results will be consistent. Otherwise, a random RNG seed will be used each time to produce a different result.

class genai.types.**GenerateVideosConfigDict**

Skip to content

Bases: `TypedDict`

Configuration for generating videos.

aspect_ratio: `Optional[str]`

16 (portrait) are supported.

TYPE:

The aspect ratio for the generated video. 16

TYPE:

9 (landscape) and 9

duration_seconds: `Optional[int]`

Duration of the clip for video generation in seconds.

enhance_prompt: `Optional[bool]`

Whether to use the prompt rewriting logic.

fps: `Optional[int]`

Frames per second for video generation.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

negative_prompt: `Optional[str]`

Optional field in addition to the text content. Negative prompts can be explicitly stated here to help generate the video.

number_of_videos: `Optional[int]`

Number of output videos.

output_gcs_uri: `Optional[str]`

The gcs bucket where to save the generated videos.

person_generation: `Optional[str]`

dont_allow, allow_adult.

TYPE:

Whether allow to generate person videos, and restrict to specific ages. Supported values are

pubsub_topic: `Optional[str]`

The pubsub topic where to publish the video generation progress.

resolution: `Optional[str]`

The resolution for the generated video. 1280x720, 1920x1080 are supported.

Skip to content

seed: `Optional[int]`

The RNG seed. If RNG seed is exactly same for each request with unchanged inputs, the prediction results will be consistent. Otherwise, a random RNG seed will be used each time to produce a different result.

pydantic model genai.types.**GenerateVideosOperation**

Skip to content

Bases: `BaseModel`

A video generation operation.

Use the following code to refresh the operation:

```
operation = client.operations.get(operation)
```

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `done (bool | None)`
- `error (dict[str, Any] | None)`
- `metadata (dict[str, Any] | None)`
- `name (str | None)`
- `response (dict[str, Any] | None)`
- `result (genai.types.GenerateVideosResponse | None)`

field done: `Optional[bool] = None`

If the value is *false*, it means the operation is still in progress. If *true*, the operation is completed, and either *error* or *response* is available.

field error: `Optional[dict[str, Any]] = None`

The error result of the operation in case of failure or cancellation.

field metadata: `Optional[dict[str, Any]] = None`

Service-specific metadata associated with the operation. It typically contains progress information and common metadata such as create time. Some services might not provide such metadata. Any method that returns a long-running operation should document the metadata type, if any.

field name: `Optional[str] = None`

The server-assigned name, which is only unique within the same service that originally returns it. If you use the default HTTP mapping, the *name* should be a resource name ending with *operations/{unique_id}*.

field response: `Optional[dict[str, Any]] = None`

The normal response of the operation in case of success.

field result: `Optional[GenerateVideosResponse] = None`

Skip to content erated videos.

class genai.types.**GenerateVideosOperationDict**

Bases: `TypedDict`

A video generation operation.

Use the following code to refresh the operation:

```
operation = client.operations.get(operation)
```

done: `Optional[bool]`

If the value is *false*, it means the operation is still in progress. If *true*, the operation is completed, and either *error* or *response* is available.

error: `Optional[dict[str, Any]]`

The error result of the operation in case of failure or cancellation.

metadata: `Optional[dict[str, Any]]`

Service-specific metadata associated with the operation. It typically contains progress information and common metadata such as create time. Some services might not provide such metadata. Any method that returns a long-running operation should document the metadata type, if any.

name: `Optional[str]`

The server-assigned name, which is only unique within the same service that originally returns it. If you use the default HTTP mapping, the *name* should be a resource name ending with *operations/{unique_id}*.

response: `Optional[dict[str, Any]]`

The normal response of the operation in case of success.

result: `Optional[GenerateVideosResponseDict]`

The generated videos.

pydantic model genai.types.GenerateVideosResponse

Bases: `BaseModel`

Response with generated videos.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

Skip to content

FIELDS:

- `generated_videos (list[genai.types.GeneratedVideo] | None)`
- `rai_media_filtered_count (int | None)`
- `rai_media_filtered_reasons (list[str] | None)`

field **generated_videos**: `Optional[list[GeneratedVideo]] = None (alias 'generatedVideos')`

List of the generated videos

field **rai_media_filtered_count**: `Optional[int] = None (alias 'raiMediaFilteredCount')`

Returns if any videos were filtered due to RAI policies.

field **rai_media_filtered_reasons**: `Optional[list[str]] = None (alias 'raiMediaFilteredReasons')`

Returns rai failure reasons if any.

class genai.types.**GenerateVideosResponseDict**

Bases: `TypedDict`

Response with generated videos.

generated_videos: `Optional[list[GeneratedVideoDict]]`

List of the generated videos

rai_media_filtered_count: `Optional[int]`

Returns if any videos were filtered due to RAI policies.

rai_media_filtered_reasons: `Optional[list[str]]`

Returns rai failure reasons if any.

pydantic model genai.types.**GeneratedImage**

Bases: `BaseModel`

An output image.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

Skip to content

FIELDS:

- `enhanced_prompt (str | None)`
- `image (genai.types.Image | None)`
- `rai_filtered_reason (str | None)`
- `safety_attributes (genai.types.SafetyAttributes | None)`

field **enhanced_prompt**: `Optional[str] = None (alias 'enhancedPrompt')`

The rewritten prompt used for the image generation if the prompt enhancer is enabled.

field **image**: `Optional[Image] = None`

The output image data.

field **rai_filtered_reason**: `Optional[str] = None (alias 'raiFilteredReason')`

Responsible AI filter reason if the image is filtered out of the response.

field **safety_attributes**: `Optional[SafetyAttributes] = None (alias 'safetyAttributes')`

Safety attributes of the image. Lists of RAI categories and their scores of each content.

class genai.types.**GeneratedImageDict**

Bases: `TypedDict`

An output image.

enhanced_prompt: `Optional[str]`

The rewritten prompt used for the image generation if the prompt enhancer is enabled.

image: `Optional[ImageDict]`

The output image data.

rai_filtered_reason: `Optional[str]`

Responsible AI filter reason if the image is filtered out of the response.

safety_attributes: `Optional[SafetyAttributesDict]`

Safety attributes of the image. Lists of RAI categories and their scores of each content.

pydantic model genai.types.**GeneratedVideo**

Bases: `BaseModel`

A generated video.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

Skip to content

ly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `video (genai.types.Video | None)`

field **video**: `Optional`[**Video**] = **None**

The output video

class genai.types.**GeneratedVideoDict**

Bases: `TypedDict`

A generated video.

video: `Optional`[**VideoDict**]

The output video

pydantic model genai.types.**GenerationConfig**

Skip to content

Bases: `BaseModel`

Generation config.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `audio_timestamp (bool | None)`
- `candidate_count (int | None)`
- `frequency_penalty (float | None)`
- `logprobs (int | None)`
- `max_output_tokens (int | None)`
- `presence_penalty (float | None)`
- `response_logprobs (bool | None)`
- `response_mime_type (str | None)`
- `response_schema (genai.types.Schema | None)`
- `routing_config (genai.types.GenerationConfigRoutingConfig | None)`
- `seed (int | None)`
- `stop_sequences (list[str] | None)`
- `temperature (float | None)`
- `top_k (float | None)`
- `top_p (float | None)`

field audio_timestamp: `Optional[bool] = None (alias 'audioTimestamp')`

Optional. If enabled, audio timestamp will be included in the request to the model.

field candidate_count: `Optional[int] = None (alias 'candidateCount')`

Optional. Number of candidates to generate.

field frequency_penalty: `Optional[float] = None (alias 'frequencyPenalty')`

Optional. Frequency penalties.

field logprobs: `Optional[int] = None`

Optional. Logit probabilities.

field max_output_tokens: `Optional[int] = None (alias 'maxOutputTokens')`

Optional. The maximum number of output tokens to generate per message.

Skip to content

field nce_penalty: `Optional[float] = None (alias 'presencePenalty')`

Optional. Positive penalties.

field **response_logprobs**: `Optional[bool] = None (alias 'responseLogprobs')`

Optional. If true, export the logprobs results in response.

field **response_mime_type**: `Optional[str] = None (alias 'responseMimeType')`

Optional. Output response mimetype of the generated candidate text. Supported mimetype: - *text/plain*: (default) Text output. - *application/json*: JSON response in the candidates. The model needs to be prompted to output the appropriate response type, otherwise the behavior is undefined. This is a preview feature.

field **response_schema**: `Optional[Schema] = None (alias 'responseSchema')`

Optional. The *Schema* object allows the definition of input and output data types. These types can be objects, but also primitives and arrays. Represents a select subset of an [OpenAPI 3.0 schema object](https://spec.openapis.org/oas/v3.0.3#schema). If set, a compatible response_mime_type must also be set. Compatible mimetypes: *application/json*: Schema for JSON response.

field **routing_config**: `Optional[GenerationConfigRoutingConfig] = None (alias 'routingConfig')`

Optional. Routing configuration.

field **seed**: `Optional[int] = None`

Optional. Seed.

field **stop_sequences**: `Optional[list[str]] = None (alias 'stopSequences')`

Optional. Stop sequences.

field **temperature**: `Optional[float] = None`

Optional. Controls the randomness of predictions.

field **top_k**: `Optional[float] = None (alias 'topK')`

Optional. If specified, top-k sampling will be used.

field **top_p**: `Optional[float] = None (alias 'topP')`

Optional. If specified, nucleus sampling will be used.

class genai.types.**GenerationConfigDict**

Bases: `TypedDict`

Generation config.

audio_timestamp: `Optional[bool]`

Optional. If enabled, audio timestamp will be included in the request to the model.

candidate_count: `Optional[int]`

Optional. Number of candidates to generate.

frequency_penalty: `Optional[float]`

Optional. Frequency penalties.

logprobs: `Optional[int]`

Optional. Logit probabilities.

max_output_tokens: `Optional[int]`

Optional. The maximum number of output tokens to generate per message.

presence_penalty: `Optional[float]`

Optional. Positive penalties.

response_logprobs: `Optional[bool]`

Optional. If true, export the logprobs results in response.

response_mime_type: `Optional[str]`

(default) Text output. - *application/json*: JSON response in the candidates. The model needs to be prompted to output the appropriate response type, otherwise the behavior is undefined. This is a preview feature.

TYPE:

Optional. Output response mimetype of the generated candidate text. Supported mimetype

TYPE:

- *text/plain*

response_schema: `Optional[SchemaDict]`

application/json: Schema for JSON response.

TYPE:

Optional. The *Schema* object allows the definition of input and output data types. These types can be objects, but also primitives and arrays. Represents a select subset of an [OpenAPI 3.0 schema object](https

TYPE:

://spec.openapis.org/oas/v3.0.3#schema). If set, a compatible response_mime_type

Skip to content t also be set. Compatible mimetypes

routing_config: `Optional[GenerationConfigRoutingConfigDict]`

Optional. Routing configuration.

seed: `Optional[int]`

Optional. Seed.

stop_sequences: `Optional[list[str]]`

Optional. Stop sequences.

temperature: `Optional[float]`

Optional. Controls the randomness of predictions.

top_k: `Optional[float]`

Optional. If specified, top-k sampling will be used.

top_p: `Optional[float]`

Optional. If specified, nucleus sampling will be used.

pydantic model genai.types.**GenerationConfigRoutingConfig**

Bases: `BaseModel`

The configuration for routing the request to a specific model.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `auto_mode (genai.types.GenerationConfigRoutingConfigAutoRoutingMode | None)`
- `manual_mode (genai.types.GenerationConfigRoutingConfigManualRoutingMode | None)`

field auto_mode: `Optional[GenerationConfigRoutingConfigAutoRoutingMode] = None (alias 'autoMode')`

Automated routing.

field manual_mode: `Optional[GenerationConfigRoutingConfigManualRoutingMode] = None (alias 'manualMode')`

Manual routing.

pydantic model genai.types.**GenerationConfigRoutingConfigAutoRoutingMode**

Bases: `BaseModel`

When automated routing is specified, the routing will be determined by the pretrained routing model and customer provided model routing preference.

Skip to content

model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `model_routing_preference (Literal['UNKNOWN', 'PRIORITIZE_QUALITY', 'BALANCED', 'PRIORITIZE_COST'] | None)`

field model_routing_preference: Optional[Literal['UNKNOWN', 'PRIORITIZE_QUALITY', 'BALANCED', 'PRIORITIZE_COST']] = None (alias 'modelRoutingPreference')

The model routing preference.

class genai.types.**GenerationConfigRoutingConfigAutoRoutingModeDict**

Bases: `TypedDict`

When automated routing is specified, the routing will be determined by the pretrained routing model and customer provided model routing preference.

model_routing_preference: Optional[Literal['UNKNOWN', 'PRIORITIZE_QUALITY', 'BALANCED', 'PRIORITIZE_COST']]

The model routing preference.

class genai.types.**GenerationConfigRoutingConfigDict**

Bases: `TypedDict`

The configuration for routing the request to a specific model.

auto_mode: Optional[GenerationConfigRoutingConfigAutoRoutingModeDict]

Automated routing.

manual_mode: Optional[GenerationConfigRoutingConfigManualRoutingModeDict]

Manual routing.

pydantic model genai.types.**GenerationConfigRoutingConfigManualRoutingMode**

Bases: `BaseModel`

When manual routing is set, the specified model will be used directly.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

Skip to content ↖ schema

FIELDS:

- `model_name (str | None)`

field model_name: `Optional[str] = None (alias 'modelName')`

The model name to use. Only the public LLM models are accepted. e.g. ‘gemini-1.5-pro-001’.

class genai.types.**GenerationConfigRoutingConfigManualRoutingModeDict**

Bases: `TypedDict`

When manual routing is set, the specified model will be used directly.

model_name: `Optional[str]`

The model name to use. Only the public LLM models are accepted. e.g. ‘gemini-1.5-pro-001’.

pydantic model genai.types.**GetBatchJobConfig**

Bases: `BaseModel`

Optional parameters.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `http_options (genai.types.HttpOptions | None)`

field http_options: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

class genai.types.**GetBatchJobConfigDict**

Bases: `TypedDict`

Optional parameters.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

pydantic model genai.types.**GetCachedContentConfig**

Bases: `BaseModel`

Optional parameters for caches.get method.

Create a new model by parsing and validating input data from keyword arguments.

Skip to content *tionError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `http_options (genai.types.HttpOptions | None)`

field **http_options**: `Optional`[**HttpOptions**] = **None** (**alias** '**httpOptions**')

Used to override HTTP request options.

class genai.types.**GetCachedContentConfigDict**

Bases: `TypedDict`

Optional parameters for caches.get method.

http_options: `Optional`[**HttpOptionsDict**]

Used to override HTTP request options.

pydantic model genai.types.**GetFileConfig**

Bases: `BaseModel`

Used to override the default configuration.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `http_options (genai.types.HttpOptions | None)`

field **http_options**: `Optional`[**HttpOptions**] = **None** (**alias** '**httpOptions**')

Used to override HTTP request options.

class genai.types.**GetFileConfigDict**

Bases: `TypedDict`

Used to override the default configuration.

http_options: `Optional`[**HttpOptionsDict**]

Used to override HTTP request options.

pydantic model genai.types.**GetModelConfig**

Bases: `BaseModel`

Skip to content meters for models.get method.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `http_options (genai.types.HttpOptions | None)`

field **http_options**: `Optional`[**HttpOptions**] = **None** (**alias** '**httpOptions**')

Used to override HTTP request options.

class genai.types.**GetModelConfigDict**

Bases: `TypedDict`

Optional parameters for models.get method.

http_options: `Optional`[**HttpOptionsDict**]

Used to override HTTP request options.

pydantic model genai.types.**GetOperationConfig**

Bases: `BaseModel`

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `http_options (genai.types.HttpOptions | None)`

field **http_options**: `Optional`[**HttpOptions**] = **None** (**alias** '**httpOptions**')

Used to override HTTP request options.

class genai.types.**GetOperationConfigDict**

Bases: `TypedDict`

http_options: `Optional`[**HttpOptionsDict**]

Used to override HTTP request options.

pydantic model genai.types.**GetTuningJobConfig**

Bases: `BaseModel`

Skip to content meters for tunings.get method.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `http_options (genai.types.HttpOptions | None)`

field **http_options**: `Optional`[**HttpOptions**] = **None** (**alias** '**httpOptions**')

Used to override HTTP request options.

class genai.types.**GetTuningJobConfigDict**

Bases: `TypedDict`

Optional parameters for tunings.get method.

http_options: `Optional`[**HttpOptionsDict**]

Used to override HTTP request options.

pydantic model genai.types.**GoogleRpcStatus**

Skip to content

Bases: `BaseModel`

The *Status* type defines a logical error model that is suitable for different programming environments, including REST APIs and RPC APIs.

It is used by [gRPC](https://github.com/grpc). Each *Status* message contains three pieces of data: error code, error message, and error details. You can find out more about this error model and how to work with it in the [API Design Guide](https://cloud.google.com/apis/design/errors).

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `code (int | None)`
- `details (list[dict[str, Any]] | None)`
- `message (str | None)`

field code: `Optional[int] = None`

The status code, which should be an enum value of google.rpc.Code.

field details: `Optional[list[dict[str, Any]]] = None`

A list of messages that carry the error details. There is a common set of message types for APIs to use.

field message: `Optional[str] = None`

A developer-facing error message, which should be in English. Any user-facing error message should be localized and sent in the google.rpc.Status.details field, or localized by the client.

class genai.types.**GoogleRpcStatusDict**

Skip to content

Bases: `TypedDict`

The *Status* type defines a logical error model that is suitable for different programming environments, including REST APIs and RPC APIs.

It is used by [gRPC](https://github.com/grpc). Each *Status* message contains three pieces of data: error code, error message, and error details. You can find out more about this error model and how to work with it in the [API Design Guide](https://cloud.google.com/apis/design/errors).

code: `Optional[int]`

The status code, which should be an enum value of google.rpc.Code.

details: `Optional[list[dict[str, Any]]]`

A list of messages that carry the error details. There is a common set of message types for APIs to use.

message: `Optional[str]`

A developer-facing error message, which should be in English. Any user-facing error message should be localized and sent in the google.rpc.Status.details field, or localized by the client.

pydantic model genai.types.**GoogleSearch**

Bases: `BaseModel`

Tool to support Google Search in Model. Powered by Google.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

class genai.types.**GoogleSearchDict**

Bases: `TypedDict`

Tool to support Google Search in Model. Powered by Google.

pydantic model genai.types.**GoogleSearchRetrieval**

Bases: `BaseModel`

Tool to retrieve public web data for grounding, powered by Google.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

Skip to content ly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `dynamic_retrieval_config (genai.types.DynamicRetrievalConfig | None)`

field **dynamic_retrieval_config**: `Optional[DynamicRetrievalConfig] = None (alias 'dynamicRetrievalConfig')`

Specifies the dynamic retrieval configuration for the given source.

class genai.types.**GoogleSearchRetrievalDict**

Bases: `TypedDict`

Tool to retrieve public web data for grounding, powered by Google.

dynamic_retrieval_config: `Optional[DynamicRetrievalConfigDict]`

Specifies the dynamic retrieval configuration for the given source.

pydantic model genai.types.**GoogleTypeDate**

Bases: `BaseModel`

Represents a whole or partial calendar date, such as a birthday.

The time of day and time zone are either specified elsewhere or are insignificant. The date is relative to the Gregorian Calendar. This can represent one of the following: * A full date, with non-zero year, month, and day values. * A month and day, with a zero year (for example, an anniversary). * A year on its own, with a zero month and a zero day. * A year and month, with a zero day (for example, a credit card expiration date). Related types: * google.type.TimeOfDay * google.type.DateTime * google.protobuf.Timestamp

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `day (int | None)`
- `month (int | None)`
- `year (int | None)`

field **day**: `Optional[int] = None`

Day of a month. Must be from 1 to 31 and valid for the year and month, or 0 to specify a year by itself or a year and month where the day isn't significant.

field **month**: `Optional[int] = None`

Month of a year. Must be from 1 to 12, or 0 to specify a year without a month and day.

Skip to content

field **year**: `Optional[int] = None`

Year of the date. Must be from 1 to 9999, or 0 to specify a date without a year.

class genai.types.**GoogleTypeDateDict**

Bases: `TypedDict`

Represents a whole or partial calendar date, such as a birthday.

The time of day and time zone are either specified elsewhere or are insignificant. The date is relative to the Gregorian Calendar. This can represent one of the following: * A full date, with non-zero year, month, and day values. * A month and day, with a zero year (for example, an anniversary). * A year on its own, with a zero month and a zero day. * A year and month, with a zero day (for example, a credit card expiration date). Related types: * google.type.TimeOfDay * google.type.DateTime * google.protobuf.Timestamp

day: `Optional[int]`

Day of a month. Must be from 1 to 31 and valid for the year and month, or 0 to specify a year by itself or a year and month where the day isn't significant.

month: `Optional[int]`

Month of a year. Must be from 1 to 12, or 0 to specify a year without a month and day.

year: `Optional[int]`

Year of the date. Must be from 1 to 9999, or 0 to specify a date without a year.

pydantic model genai.types.**GroundingChunk**

Bases: `BaseModel`

Grounding chunk.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `retrieved_context (genai.types.GroundingChunkRetrievedContext | None)`
- `web (genai.types.GroundingChunkWeb | None)`

field retrieved_context: `Optional[GroundingChunkRetrievedContext] = None (alias 'retrievedContext')`

Grounding chunk from context retrieved by the retrieval tools.

field web: `Optional[GroundingChunkWeb] = None`

Grounding chunk from the web.

Skip to content

class genai.types.**GroundingChunkDict**

Bases: `TypedDict`

Grounding chunk.

retrieved_context: Optional[GroundingChunkRetrievedContextDict]

Grounding chunk from context retrieved by the retrieval tools.

web: Optional[GroundingChunkWebDict]

Grounding chunk from the web.

pydantic model genai.types.**GroundingChunkRetrievedContext**

Bases: BaseModel

Chunk from context retrieved by the retrieval tools.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- text (str | None)
- title (str | None)
- uri (str | None)

field text: Optional[str] = None

Text of the attribution.

field title: Optional[str] = None

Title of the attribution.

field uri: Optional[str] = None

URI reference of the attribution.

class genai.types.**GroundingChunkRetrievedContextDict**

Bases: TypedDict

Chunk from context retrieved by the retrieval tools.

text: Optional[str]

Text of the attribution.

title: Optional[str]

Title of the attribution.

Skip to content

l[str]

ence of the attribution.

pydantic model genai.types.**GroundingChunkWeb**

Bases: `BaseModel`

Chunk from the web.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `title (str | None)`
- `uri (str | None)`

field title: `Optional[str] = None`

Title of the chunk.

field uri: `Optional[str] = None`

URI reference of the chunk.

class genai.types.**GroundingChunkWebDict**

Bases: `TypedDict`

Chunk from the web.

title: `Optional[str]`

Title of the chunk.

uri: `Optional[str]`

URI reference of the chunk.

pydantic model genai.types.**GroundingMetadata**

Bases: `BaseModel`

Metadata returned to client when grounding is enabled.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

Skip to content

FIELDS:

- `grounding_chunks (list[genai.types.GroundingChunk] | None)`
- `grounding_supports (list[genai.types.GroundingSupport] | None)`
- `retrieval_metadata (genai.types.RetrievalMetadata | None)`
- `retrieval_queries (list[str] | None)`
- `search_entry_point (genai.types.SearchEntryPoint | None)`
- `web_search_queries (list[str] | None)`

field grounding_chunks: `Optional[list[GroundingChunk]] = None (alias 'groundingChunks')`

List of supporting references retrieved from specified grounding source.

field grounding_supports: `Optional[list[GroundingSupport]] = None (alias 'groundingSupports')`

Optional. List of grounding support.

field retrieval_metadata: `Optional[RetrievalMetadata] = None (alias 'retrievalMetadata')`

Optional. Output only. Retrieval metadata.

field retrieval_queries: `Optional[list[str]] = None (alias 'retrievalQueries')`

Optional. Queries executed by the retrieval tools.

field search_entry_point: `Optional[SearchEntryPoint] = None (alias 'searchEntryPoint')`

Optional. Google search entry for the following-up web searches.

field web_search_queries: `Optional[list[str]] = None (alias 'webSearchQueries')`

Optional. Web search queries for the following-up web search.

class genai.types.**GroundingMetadataDict**

Skip to content

Bases: `TypedDict`

Metadata returned to client when grounding is enabled.

grounding_chunks: `Optional[list[GroundingChunkDict]]`

List of supporting references retrieved from specified grounding source.

grounding_supports: `Optional[list[GroundingSupportDict]]`

Optional. List of grounding support.

retrieval_metadata: `Optional[RetrievalMetadataDict]`

Optional. Output only. Retrieval metadata.

retrieval_queries: `Optional[list[str]]`

Optional. Queries executed by the retrieval tools.

search_entry_point: `Optional[SearchEntryPointDict]`

Optional. Google search entry for the following-up web searches.

web_search_queries: `Optional[list[str]]`

Optional. Web search queries for the following-up web search.

pydantic model genai.types.**GroundingSupport**

Skip to content

Bases: `BaseModel`

Grounding support.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `confidence_scores (list[float] | None)`
- `grounding_chunk_indices (list[int] | None)`
- `segment (genai.types.Segment | None)`

field **confidence_scores**: `Optional[list[float]] = None (alias 'confidenceScores')`

Confidence score of the support references. Ranges from 0 to 1. 1 is the most confident. This list must have the same size as the grounding_chunk_indices.

field **grounding_chunk_indices**: `Optional[list[int]] = None (alias 'groundingChunkIndices')`

A list of indices (into 'grounding_chunk') specifying the citations associated with the claim. For instance [1,3,4] means that grounding_chunk[1], grounding_chunk[3], grounding_chunk[4] are the retrieved content attributed to the claim.

field **segment**: `Optional[Segment] = None`

Segment of the content this support belongs to.

class genai.types.**GroundingSupportDict**

Bases: `TypedDict`

Grounding support.

confidence_scores: `Optional[list[float]]`

Confidence score of the support references. Ranges from 0 to 1. 1 is the most confident. This list must have the same size as the grounding_chunk_indices.

grounding_chunk_indices: `Optional[list[int]]`

A list of indices (into 'grounding_chunk') specifying the citations associated with the claim. For instance [1,3,4] means that grounding_chunk[1], grounding_chunk[3], grounding_chunk[4] are the retrieved content attributed to the claim.

segment: `Optional[SegmentDict]`

Segment of the content this support belongs to.

Skip to content

class genai.types.**HarmBlockMethod**(*values)

Bases: `CaseInSensitiveEnum`

Optional.

Specify if the threshold is used for probability or severity score. If not specified, the threshold is used for probability score.

```
HARM_BLOCK_METHOD_UNSPECIFIED = 'HARM_BLOCK_METHOD_UNSPECIFIED'
PROBABILITY = 'PROBABILITY'
SEVERITY = 'SEVERITY'
```

class genai.types.**HarmBlockThreshold**(*values)

Bases: `CaseInSensitiveEnum`

Required. The harm block threshold.

```
BLOCK_LOW_AND_ABOVE = 'BLOCK_LOW_AND_ABOVE'
BLOCK_MEDIUM_AND_ABOVE = 'BLOCK_MEDIUM_AND_ABOVE'
BLOCK_NONE = 'BLOCK_NONE'
BLOCK_ONLY_HIGH = 'BLOCK_ONLY_HIGH'
HARM_BLOCK_THRESHOLD_UNSPECIFIED = 'HARM_BLOCK_THRESHOLD_UNSPECIFIED'
OFF = 'OFF'
```

class genai.types.**HarmCategory**(*values)

Bases: `CaseInSensitiveEnum`

Required. Harm category.

```
HARM_CATEGORY_CIVIC_INTEGRITY = 'HARM_CATEGORY_CIVIC_INTEGRITY'
HARM_CATEGORY_DANGEROUS_CONTENT = 'HARM_CATEGORY_DANGEROUS_CONTENT'
HARM_CATEGORY_HARASSMENT = 'HARM_CATEGORY_HARASSMENT'
HARM_CATEGORY_HATE_SPEECH = 'HARM_CATEGORY_HATE_SPEECH'
HARM_CATEGORY_SEXUALLY_EXPLICIT = 'HARM_CATEGORY_SEXUALLY_EXPLICIT'
HARM_CATEGORY_UNSPECIFIED = 'HARM_CATEGORY_UNSPECIFIED'
```

class genai.types.**HarmProbability**(*values)

Bases: `CaseInSensitiveEnum`

Output only. Harm probability levels in the content.

```
HARM_PROBABILITY_UNSPECIFIED = 'HARM_PROBABILITY_UNSPECIFIED'
HIGH = 'HIGH'
LOW = 'LOW'
MEDIUM = 'MEDIUM'
           = 'NEGLIGIBLE'
```

Skip to content

class genai.types.**HarmSeverity**(*values)

Bases: `CaseInSensitiveEnum`

Output only. Harm severity levels in the content.

```
HARM_SEVERITY_HIGH = 'HARM_SEVERITY_HIGH'
HARM_SEVERITY_LOW = 'HARM_SEVERITY_LOW'
HARM_SEVERITY_MEDIUM = 'HARM_SEVERITY_MEDIUM'
HARM_SEVERITY_NEGLIGIBLE = 'HARM_SEVERITY_NEGLIGIBLE'
HARM_SEVERITY_UNSPECIFIED = 'HARM_SEVERITY_UNSPECIFIED'
```

pydantic model genai.types.**HttpOptions**

Bases: `BaseModel`

HTTP options to be used in each of the requests.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `api_version (str | None)`
- `base_url (str | None)`
- `headers (dict[str, str] | None)`
- `timeout (int | None)`

field api_version: `Optional[str] = None (alias 'apiVersion')`

Specifies the version of the API to use.

field base_url: `Optional[str] = None (alias 'baseUrl')`

The base URL for the AI platform service endpoint.

field headers: `Optional[dict[str, str]] = None`

Additional HTTP headers to be sent with the request.

field timeout: `Optional[int] = None`

Timeout for the request in milliseconds.

class genai.types.**HttpOptionsDict**

Skip to content

Bases: `TypedDict`

HTTP options to be used in each of the requests.

api_version: `Optional[str]`

Specifies the version of the API to use.

base_url: `Optional[str]`

The base URL for the AI platform service endpoint.

headers: `Optional[dict[str, str]]`

Additional HTTP headers to be sent with the request.

timeout: `Optional[int]`

Timeout for the request in milliseconds.

pydantic model genai.types.**Image**

Skip to content

Bases: `BaseModel`

An image.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `gcs_uri (str | None)`
- `image_bytes (bytes | None)`
- `mime_type (str | None)`

field gcs_uri: `Optional[str] = None (alias 'gcsUri')`

The Cloud Storage URI of the image. `Image` can contain a value for this field or the `image_bytes` field but not both.

field image_bytes: `Optional[bytes] = None (alias 'imageBytes')`

The image bytes data. `Image` can contain a value for this field or the `gcs_uri` field but not both.

field mime_type: `Optional[str] = None (alias 'mimeType')`

The MIME type of the image.

classmethod from_file(*, location, mime_type=None)

Lazy-loads an image from a local file or Google Cloud Storage.

RETURN TYPE:

`Image`

PARAMETERS:

- **location** – The local path or Google Cloud Storage URI from which to load the image.
- **mime_type** – The MIME type of the image. If not provided, the MIME type will be automatically determined.

RETURNS:

A loaded image as an *Image* object.

model_post_init(context, /)

This function is meant to behave like a BaseModel method to initialise private attributes.

Skip to content

It takes context as an argument since that's what pydantic-core passes when calling it.

RETURN TYPE:

None

PARAMETERS:

- **self** – The BaseModel instance.
- **context** – The context.

`save(location)`

Saves the image to a file.

PARAMETERS:

location – Local path where to save the image.

`show()`

Shows the image.

This method only works in a notebook environment.

`class genai.types.ImageDict`

Bases: `TypedDict`

An image.

`gcs_uri: Optional[str]`

The Cloud Storage URI of the image. `Image` can contain a value for this field or the `image_bytes` field but not both.

`image_bytes: Optional[bytes]`

The image bytes data. `Image` can contain a value for this field or the `gcs_uri` field but not both.

`mime_type: Optional[str]`

The MIME type of the image.

`class genai.types.ImagePromptLanguage(*values)`

Bases: `CaseInSensitiveEnum`

Enum that specifies the language of the text in the prompt.

`auto = 'auto'`

`en = 'en'`

`hi = 'hi'`

`ja = 'ja'`

`ko = 'ko'`

Skip to content

`genai.types.JobError`

Bases: `BaseModel`

Job error.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `code (int | None)`
- `details (list[str] | None)`
- `message (str | None)`

field code: `Optional[int] = None`

The status code.

field details: `Optional[list[str]] = None`

A list of messages that carry the error details. There is a common set of message types for APIs to use.

field message: `Optional[str] = None`

A developer-facing error message, which should be in English. Any user-facing error message should be localized and sent in the *details* field.

class genai.types.**JobErrorDict**

Bases: `TypedDict`

Job error.

code: `Optional[int]`

The status code.

details: `Optional[list[str]]`

A list of messages that carry the error details. There is a common set of message types for APIs to use.

message: `Optional[str]`

A developer-facing error message, which should be in English. Any user-facing error message should be localized and sent in the *details* field.

class genai.types.**JobState**(*values)

Skip to content

Bases: `CaseInSensitiveEnum`

Job state.

```
JOB_STATE_CANCELLED = 'JOB_STATE_CANCELLED'
JOB_STATE_CANCELLING = 'JOB_STATE_CANCELLING'
JOB_STATE_EXPIRED = 'JOB_STATE_EXPIRED'
JOB_STATE_FAILED = 'JOB_STATE_FAILED'
JOB_STATE_PARTIALLY_SUCCEEDED = 'JOB_STATE_PARTIALLY_SUCCEEDED'
JOB_STATE_PAUSED = 'JOB_STATE_PAUSED'
JOB_STATE_PENDING = 'JOB_STATE_PENDING'
JOB_STATE_QUEUED = 'JOB_STATE_QUEUED'
JOB_STATE_RUNNING = 'JOB_STATE_RUNNING'
JOB_STATE_SUCCEEDED = 'JOB_STATE_SUCCEEDED'
JOB_STATE_UNSPECIFIED = 'JOB_STATE_UNSPECIFIED'
JOB_STATE_UPDATING = 'JOB_STATE_UPDATING'
```

class genai.types.**Language**(*values)

Bases: `CaseInSensitiveEnum`

Required. Programming language of the *code*.

```
LANGUAGE_UNSPECIFIED = 'LANGUAGE_UNSPECIFIED'
PYTHON = 'PYTHON'
```

pydantic model genai.types.**ListBatchJobsConfig**

Skip to content

Bases: `BaseModel`

Config for optional parameters.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `filter (str | None)`
- `http_options (genai.types.HttpOptions | None)`
- `page_size (int | None)`
- `page_token (str | None)`

field filter: `Optional[str] = None`

field http_options: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

field page_size: `Optional[int] = None (alias 'pageSize')`

field page_token: `Optional[str] = None (alias 'pageToken')`

class genai.types.**ListBatchJobsConfigDict**

Bases: `TypedDict`

Config for optional parameters.

filter: `Optional[str]`

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

page_size: `Optional[int]`

page_token: `Optional[str]`

pydantic model genai.types.**ListBatchJobsResponse**

Bases: `BaseModel`

Config for batches.list return value.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

Skip to content

ly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `batch_jobs (list[genai.types.BatchJob] | None)`
- `next_page_token (str | None)`

field batch_jobs: `Optional[list[BatchJob]] = None (alias 'batchJobs')`

field next_page_token: `Optional[str] = None (alias 'nextPageToken')`

class genai.types.**ListBatchJobsResponseDict**

Bases: `TypedDict`

Config for batches.list return value.

batch_jobs: `Optional[list[BatchJobDict]]`

next_page_token: `Optional[str]`

pydantic model genai.types.**ListCachedContentsConfig**

Bases: `BaseModel`

Config for caches.list method.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `http_options (genai.types.HttpOptions | None)`
- `page_size (int | None)`
- `page_token (str | None)`

field http_options: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

field page_size: `Optional[int] = None (alias 'pageSize')`

field page_token: `Optional[str] = None (alias 'pageToken')`

class genai.types.**ListCachedContentsConfigDict**

Bases: `TypedDict`

Config for caches.list method.

http_options: `Optional[HttpOptionsDict]`

override HTTP request options.

Skip to content

page_size: `Optional[int]`

page_token: `Optional[str]`

pydantic model genai.types.**ListCachedContentsResponse**

Bases: `BaseModel`

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `cached_contents (list[genai.types.CachedContent] | None)`
- `next_page_token (str | None)`

field cached_contents: `Optional[list[CachedContent]] = None (alias 'cachedContents')`

List of cached contents.

field next_page_token: `Optional[str] = None (alias 'nextPageToken')`

class genai.types.**ListCachedContentsResponseDict**

Bases: `TypedDict`

cached_contents: `Optional[list[CachedContentDict]]`

List of cached contents.

next_page_token: `Optional[str]`

pydantic model genai.types.**ListFilesConfig**

Skip to content

Bases: `BaseModel`

Used to override the default configuration.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `http_options (genai.types.HttpOptions | None)`
- `page_size (int | None)`
- `page_token (str | None)`

field **http_options**: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

field **page_size**: `Optional[int] = None (alias 'pageSize')`

field **page_token**: `Optional[str] = None (alias 'pageToken')`

class genai.types.**ListFilesConfigDict**

Bases: `TypedDict`

Used to override the default configuration.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

page_size: `Optional[int]`

page_token: `Optional[str]`

pydantic model genai.types.**ListFilesResponse**

Bases: `BaseModel`

Response for the list files method.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

Skip to content

FIELDS:

- `files (list[genai.types.File] | None)`
- `next_page_token (str | None)`

field files: `Optional[list[File]] = None`

The list of files.

field next_page_token: `Optional[str] = None (alias 'nextPageToken')`

A token to retrieve next page of results.

class genai.types.**ListFilesResponseDict**

Bases: `TypedDict`

Response for the list files method.

files: `Optional[list[FileDict]]`

The list of files.

next_page_token: `Optional[str]`

A token to retrieve next page of results.

pydantic model genai.types.**ListModelsConfig**

Bases: `BaseModel`

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `filter (str | None)`
- `http_options (genai.types.HttpOptions | None)`
- `page_size (int | None)`
- `page_token (str | None)`
- `query_base (bool | None)`

field filter: `Optional[str] = None`

field http_options: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

field page_size: `Optional[int] = None (alias 'pageSize')`

token: `Optional[str] = None (alias 'pageToken')`

_base: `Optional[bool] = None (alias 'queryBase')`

Set true to list base models, false to list tuned models.

Skip to content

class genai.types.**ListModelsConfigDict**

Bases: `TypedDict`

filter: `Optional[str]`

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

page_size: `Optional[int]`

page_token: `Optional[str]`

query_base: `Optional[bool]`

Set true to list base models, false to list tuned models.

pydantic model genai.types.**ListModelsResponse**

Bases: `BaseModel`

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `models (list[genai.types.Model] | None)`
- `next_page_token (str | None)`

field models: `Optional[list[Model]] = None`

field next_page_token: `Optional[str] = None (alias 'nextPageToken')`

class genai.types.**ListModelsResponseDict**

Bases: `TypedDict`

models: `Optional[list[ModelDict]]`

next_page_token: `Optional[str]`

pydantic model genai.types.**ListTuningJobsConfig**

Bases: `BaseModel`

Configuration for the list tuning jobs method.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

Skip to content

ly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `filter (str | None)`
- `http_options (genai.types.HttpOptions | None)`
- `page_size (int | None)`
- `page_token (str | None)`

field filter: `Optional[str] = None`

field http_options: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

field page_size: `Optional[int] = None (alias 'pageSize')`

field page_token: `Optional[str] = None (alias 'pageToken')`

class genai.types.**ListTuningJobsConfigDict**

Bases: `TypedDict`

Configuration for the list tuning jobs method.

filter: `Optional[str]`

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

page_size: `Optional[int]`

page_token: `Optional[str]`

pydantic model genai.types.**ListTuningJobsResponse**

Bases: `BaseModel`

Response for the list tuning jobs method.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `next_page_token (str | None)`
- `tuning_jobs (list[genai.types.TuningJob] | None)`

field next_page_token: `Optional[str] = None (alias 'nextPageToken')`

A token to retrieve the next page of results. Pass to ListTuningJobsRequest.page_token to `at page.

Skip to content

field tuning_jobs: `Optional[list[TuningJob]] = None (alias 'tuningJobs')`

List of TuningJobs in the requested page.

class genai.types.**ListTuningJobsResponseDict**

Bases: `TypedDict`

Response for the list tuning jobs method.

next_page_token: `Optional[str]`

A token to retrieve the next page of results. Pass to ListTuningJobsRequest.page_token to obtain that page.

tuning_jobs: `Optional[list[TuningJobDict]]`

List of TuningJobs in the requested page.

pydantic model genai.types.**LiveClientContent**

Bases: `BaseModel`

Incremental update of the current conversation delivered from the client.

All the content here will unconditionally be appended to the conversation history and used as part of the prompt to the model to generate content.

A message here will interrupt any current model generation.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `turn_complete (bool | None)`
- `turns (list[genai.types.Content] | None)`

field turn_complete: `Optional[bool] = None (alias 'turnComplete')`

If true, indicates that the server content generation should start with the currently accumulated prompt. Otherwise, the server will await additional messages before starting generation.

field turns: `Optional[list[Content]] = None`

The content appended to the current conversation with the model.

For single-turn queries, this is a single instance. For multi-turn queries, this is a repeated field that contains conversation history and latest request.

class genai.types.**LiveClientContentDict**

Bases: `TypedDict`

Incremental update of the current conversation delivered from the client.

Skip to content

All the content here will unconditionally be appended to the conversation history and used as part of the prompt to the model to generate content.

A message here will interrupt any current model generation.

turn_complete: Optional[bool]

If true, indicates that the server content generation should start with the currently accumulated prompt. Otherwise, the server will await additional messages before starting generation.

turns: Optional[list[ContentDict]]

The content appended to the current conversation with the model.

For single-turn queries, this is a single instance. For multi-turn queries, this is a repeated field that contains conversation history and latest request.

pydantic model genai.types.**LiveClientMessage**

Bases: BaseModel

Messages sent by the client in the API call.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- client_content (genai.types.LiveClientContent | None)
- realtime_input (genai.types.LiveClientRealtimeInput | None)
- setup (genai.types.LiveClientSetup | None)
- tool_response (genai.types.LiveClientToolResponse | None)

field client_content: Optional[LiveClientContent] = None (alias 'clientContent')

Incremental update of the current conversation delivered from the client.

field realtime_input: Optional[LiveClientRealtimeInput] = None (alias 'realtimeInput')

User input that is sent in real time.

field setup: Optional[LiveClientSetup] = None

Message to be sent by the system when connecting to the API. SDK users should not send this message.

field tool_response: Optional[LiveClientToolResponse] = None (alias 'toolResponse')

Skip to content e to a *ToolCallMessage* received from the server.

class genai.types.**LiveClientMessageDict**

Bases: `TypedDict`

Messages sent by the client in the API call.

client_content: `Optional[LiveClientContentDict]`

Incremental update of the current conversation delivered from the client.

realtime_input: `Optional[LiveClientRealtimeInputDict]`

User input that is sent in real time.

setup: `Optional[LiveClientSetupDict]`

Message to be sent by the system when connecting to the API. SDK users should not send this message.

tool_response: `Optional[LiveClientToolResponseDict]`

Response to a *ToolCallMessage* received from the server.

pydantic model genai.types.LiveClientRealtimeInput

Bases: `BaseModel`

User input that is sent in real time.

This is different from *ClientContentUpdate* in a few ways:

> - Can be sent continuously without interruption to model generation.
> - If there is a need to mix data interleaved across the *ClientContentUpdate* and the *RealtimeUpdate*, server attempts to optimize for best response, but there are no guarantees.
> - End of turn is not explicitly specified, but is rather derived from user activity (for example, end of speech).
> - Even before the end of turn, the data is processed incrementally to optimize for a fast start of the response from the model.
> - Is always assumed to be the user's input (cannot be used to populate conversation history).

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

Skip to content

FIELDS:

- `media_chunks (list[genai.types.Blob] | None)`

field media_chunks: `Optional[list[Blob]] = None (alias 'mediaChunks')`

Inlined bytes data for media input.

class genai.types.**LiveClientRealtimeInputDict**

Bases: `TypedDict`

User input that is sent in real time.

This is different from *ClientContentUpdate* in a few ways:

> - Can be sent continuously without interruption to model generation.
> - If there is a need to mix data interleaved across the *ClientContentUpdate* and the *RealtimeUpdate*, server attempts to optimize for best response, but there are no guarantees.
> - End of turn is not explicitly specified, but is rather derived from user activity (for example, end of speech).
> - Even before the end of turn, the data is processed incrementally to optimize for a fast start of the response from the model.
> - Is always assumed to be the user's input (cannot be used to populate conversation history).

media_chunks: `Optional[list[BlobDict]]`

Inlined bytes data for media input.

pydantic model genai.types.**LiveClientSetup**

Bases: `BaseModel`

Message contains configuration that will apply for the duration of the streaming session.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

Skip to content

FIELDS:

- `generation_config (genai.types.GenerationConfig | None)`
- `model (str | None)`
- `system_instruction (genai.types.Content | None)`
- `tools (list[genai.types.Tool | Callable] | None)`

field generation_config: Optional[GenerationConfig] = None (alias 'generationConfig')

The generation configuration for the session.

The following fields are supported: - *response_logprobs* - *response_mime_type* - *logprobs* - *response_schema* - *stop_sequence* - *routing_config* - *audio_timestamp*

field model: Optional[str] = None

The fully qualified name of the publisher model or tuned model endpoint to use.

field system_instruction: Optional[Content] = None (alias 'systemInstruction')

The user provided system instructions for the model. Note: only text should be used in parts and content in each part will be in a separate paragraph.

field tools: Optional[list[Union[Tool, Callable]]] = None

A list of *Tools* the model may use to generate the next response.

A *Tool* is a piece of code that enables the system to interact with external systems to perform an action, or set of actions, outside of knowledge and scope of the model.

class genai.types.LiveClientSetupDict

Bases: `TypedDict`

Message contains configuration that will apply for the duration of the streaming session.

generation_config: Optional[GenerationConfigDict]

The generation configuration for the session.

The following fields are supported: - *response_logprobs* - *response_mime_type* - *logprobs* - *response_schema* - *stop_sequence* - *routing_config* - *audio_timestamp*

model: Optional[str]

The fully qualified name of the publisher model or tuned model endpoint to use.

system_instruction: Optional[ContentDict]

The user provided system instructions for the model. Note: only text should be used in parts and content in each part will be in a separate paragraph.

tools: Optional[list[Union[ToolDict, Callable]]]

Tools the model may use to generate the next response.

Skip to content

, a piece of code that enables the system to interact with external systems to perform an action, or set of actions, outside of knowledge and scope of the model.

pydantic model genai.types.**LiveClientToolResponse**

Bases: `BaseModel`

Client generated response to a *ToolCall* received from the server.

Individual *FunctionResponse* objects are matched to the respective *FunctionCall* objects by the *id* field.

Note that in the unary and server-streaming GenerateContent APIs function calling happens by exchanging the *Content* parts, while in the bidi GenerateContent APIs function calling happens over this dedicated set of messages.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `function_responses (list[genai.types.FunctionResponse] | None)`

field function_responses: `Optional[list[FunctionResponse]] = None (alias 'functionResponses')`

The response to the function calls.

class genai.types.**LiveClientToolResponseDict**

Bases: `TypedDict`

Client generated response to a *ToolCall* received from the server.

Individual *FunctionResponse* objects are matched to the respective *FunctionCall* objects by the *id* field.

Note that in the unary and server-streaming GenerateContent APIs function calling happens by exchanging the *Content* parts, while in the bidi GenerateContent APIs function calling happens over this dedicated set of messages.

function_responses: `Optional[list[FunctionResponseDict]]`

The response to the function calls.

pydantic model genai.types.**LiveConnectConfig**

Bases: `BaseModel`

Session config for the API connection.

Create a new model by parsing and validating input data from keyword arguments.

Skip to content

tionError][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `generation_config (genai.types.GenerationConfig | None)`
- `response_modalities (list[genai.types.Modality] | None)`
- `speech_config (genai.types.SpeechConfig | None)`
- `system_instruction (genai.types.Content | None)`
- `tools (list[genai.types.Tool | Callable] | None)`

field **generation_config**: `Optional[GenerationConfig] = None (alias 'generationConfig')`

The generation configuration for the session.

field **response_modalities**: `Optional[list[Modality]] = None (alias 'responseModalities')`

The requested modalities of the response. Represents the set of modalities that the model can return. Defaults to AUDIO if not specified.

field **speech_config**: `Optional[SpeechConfig] = None (alias 'speechConfig')`

The speech generation configuration.

field **system_instruction**: `Optional[Content] = None (alias 'systemInstruction')`

The user provided system instructions for the model. Note: only text should be used in parts and content in each part will be in a separate paragraph.

field **tools**: `Optional[list[Union[Tool, Callable]]] = None`

A list of *Tools* the model may use to generate the next response.

A *Tool* is a piece of code that enables the system to interact with external systems to perform an action, or set of actions, outside of knowledge and scope of the model.

class genai.types.**LiveConnectConfigDict**

Skip to content

Bases: `TypedDict`

Session config for the API connection.

generation_config: `Optional[GenerationConfigDict]`

The generation configuration for the session.

response_modalities: `Optional[list[Modality]]`

The requested modalities of the response. Represents the set of modalities that the model can return. Defaults to AUDIO if not specified.

speech_config: `Optional[SpeechConfigDict]`

The speech generation configuration.

system_instruction: `Optional[ContentDict]`

The user provided system instructions for the model. Note: only text should be used in parts and content in each part will be in a separate paragraph.

tools: `Optional[list[Union[ToolDict, Callable]]]`

A list of *Tools* the model may use to generate the next response.

A *Tool* is a piece of code that enables the system to interact with external systems to perform an action, or set of actions, outside of knowledge and scope of the model.

pydantic model genai.types.**LiveServerContent**

Skip to content

Bases: `BaseModel`

Incremental server update generated by the model in response to client messages.

Content is generated as quickly as possible, and not in real time. Clients may choose to buffer and play it out in real time.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `interrupted (bool | None)`
- `model_turn (genai.types.Content | None)`
- `turn_complete (bool | None)`

field interrupted: `Optional[bool] = None`

If true, indicates that a client message has interrupted current model generation. If the client is playing out the content in realtime, this is a good signal to stop and empty the current queue.

field model_turn: `Optional[Content] = None (alias 'modelTurn')`

The content that the model has generated as part of the current conversation with the user.

field turn_complete: `Optional[bool] = None (alias 'turnComplete')`

If true, indicates that the model is done generating. Generation will only start in response to additional client messages. Can be set alongside *content*, indicating that the *content* is the last in the turn.

class genai.types.**LiveServerContentDict**

Skip to content

Bases: `TypedDict`

Incremental server update generated by the model in response to client messages.

Content is generated as quickly as possible, and not in real time. Clients may choose to buffer and play it out in real time.

interrupted: `Optional[bool]`

If true, indicates that a client message has interrupted current model generation. If the client is playing out the content in realtime, this is a good signal to stop and empty the current queue.

model_turn: `Optional[ContentDict]`

The content that the model has generated as part of the current conversation with the user.

turn_complete: `Optional[bool]`

If true, indicates that the model is done generating. Generation will only start in response to additional client messages. Can be set alongside *content*, indicating that the *content* is the last in the turn.

pydantic model genai.types.**LiveServerMessage**

Skip to content

Bases: `BaseModel`

Response message for API call.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `server_content (genai.types.LiveServerContent | None)`
- `setup_complete (genai.types.LiveServerSetupComplete | None)`
- `tool_call (genai.types.LiveServerToolCall | None)`
- `tool_call_cancellation (genai.types.LiveServerToolCallCancellation | None)`

field server_content: `Optional[LiveServerContent] = None (alias 'serverContent')`

Content generated by the model in response to client messages.

field setup_complete: `Optional[LiveServerSetupComplete] = None (alias 'setupComplete')`

Sent in response to a *LiveClientSetup* message from the client.

field tool_call: `Optional[LiveServerToolCall] = None (alias 'toolCall')`

Request for the client to execute the *function_calls* and return the responses with the matching `id`s.

field tool_call_cancellation: `Optional[LiveServerToolCallCancellation] = None (alias 'toolCallCancellation')`

Notification for the client that a previously issued *ToolCallMessage* with the specified `id`s should have been not executed and should be cancelled.

property data: bytes | None

Returns the concatenation of all inline data parts in the response.

property text: str | None

Returns the concatenation of all text parts in the response.

class genai.types.LiveServerMessageDict

Skip to content

Bases: `TypedDict`

Response message for API call.

server_content: `Optional[LiveServerContentDict]`

Content generated by the model in response to client messages.

setup_complete: `Optional[LiveServerSetupCompleteDict]`

Sent in response to a *LiveClientSetup* message from the client.

tool_call: `Optional[LiveServerToolCallDict]`

Request for the client to execute the *function_calls* and return the responses with the matching `id`s.

tool_call_cancellation: `Optional[LiveServerToolCallCancellationDict]`

Notification for the client that a previously issued *ToolCallMessage* with the specified `id`s should have been not executed and should be cancelled.

pydantic model genai.types.LiveServerSetupComplete

Bases: `BaseModel`

Sent in response to a *LiveGenerateContentSetup* message from the client.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

class genai.types.LiveServerSetupCompleteDict

Bases: `TypedDict`

Sent in response to a *LiveGenerateContentSetup* message from the client.

pydantic model genai.types.LiveServerToolCall

Bases: `BaseModel`

Request for the client to execute the *function_calls* and return the responses with the matching `id`s.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

Skip to content

▶ Show JSON schema

FIELDS:

- `function_calls (list[genai.types.FunctionCall] | None)`

field function_calls: `Optional[list[FunctionCall]] = None (alias 'functionCalls')`

The function call to be executed.

pydantic model genai.types.**LiveServerToolCallCancellation**

Bases: `BaseModel`

Notification for the client that a previously issued *ToolCallMessage* with the specified `id`s should have been not executed and should be cancelled.

If there were side-effects to those tool calls, clients may attempt to undo the tool calls. This message occurs only in cases where the clients interrupt server turns.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `ids (list[str] | None)`

field ids: `Optional[list[str]] = None`

The ids of the tool calls to be cancelled.

class genai.types.**LiveServerToolCallCancellationDict**

Bases: `TypedDict`

Notification for the client that a previously issued *ToolCallMessage* with the specified `id`s should have been not executed and should be cancelled.

If there were side-effects to those tool calls, clients may attempt to undo the tool calls. This message occurs only in cases where the clients interrupt server turns.

ids: `Optional[list[str]]`

The ids of the tool calls to be cancelled.

class genai.types.**LiveServerToolCallDict**

Bases: `TypedDict`

Request for the client to execute the *function_calls* and return the responses with the matching `id`s.

Skip to content

lls: `Optional[list[FunctionCallDict]]`

tion call to be executed.

pydantic model genai.types.**LogprobsResult**

Bases: `BaseModel`

Logprobs Result

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `chosen_candidates (list[genai.types.LogprobsResultCandidate] | None)`
- `top_candidates (list[genai.types.LogprobsResultTopCandidates] | None)`

field chosen_candidates: `Optional[list[LogprobsResultCandidate]] = None (alias 'chosenCandidates')`

Length = total number of decoding steps. The chosen candidates may or may not be in top_candidates.

field top_candidates: `Optional[list[LogprobsResultTopCandidates]] = None (alias 'topCandidates')`

Length = total number of decoding steps.

pydantic model genai.types.**LogprobsResultCandidate**

Skip to content

Bases: `BaseModel`

Candidate for the logprobs token and score.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `log_probability (float | None)`
- `token (str | None)`
- `token_id (int | None)`

field log_probability: `Optional[float] = None (alias 'logProbability')`

The candidate's log probability.

field token: `Optional[str] = None`

The candidate's token string value.

field token_id: `Optional[int] = None (alias 'tokenId')`

The candidate's token id value.

class genai.types.**LogprobsResultCandidateDict**

Bases: `TypedDict`

Candidate for the logprobs token and score.

log_probability: `Optional[float]`

The candidate's log probability.

token: `Optional[str]`

The candidate's token string value.

token_id: `Optional[int]`

The candidate's token id value.

class genai.types.**LogprobsResultDict**

Skip to content

Bases: `TypedDict`

Logprobs Result

chosen_candidates: `Optional[list[LogprobsResultCandidateDict]]`

Length = total number of decoding steps. The chosen candidates may or may not be in top_candidates.

top_candidates: `Optional[list[LogprobsResultTopCandidatesDict]]`

Length = total number of decoding steps.

pydantic model genai.types.**LogprobsResultTopCandidates**

Bases: `BaseModel`

Candidates with top log probabilities at each decoding step.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `candidates (list[genai.types.LogprobsResultCandidate] | None)`

field candidates: `Optional[list[LogprobsResultCandidate]] = None`

Sorted by log probability in descending order.

class genai.types.**LogprobsResultTopCandidatesDict**

Bases: `TypedDict`

Candidates with top log probabilities at each decoding step.

candidates: `Optional[list[LogprobsResultCandidateDict]]`

Sorted by log probability in descending order.

pydantic model genai.types.**MaskReferenceConfig**

Bases: `BaseModel`

Configuration for a Mask reference image.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

Skip to content ly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `mask_dilation (float | None)`
- `mask_mode (genai.types.MaskReferenceMode | None)`
- `segmentation_classes (list[int] | None)`

field mask_dilation: `Optional[float] = None (alias 'maskDilation')`

Dilation percentage of the mask provided. Float between 0 and 1.

field mask_mode: `Optional[MaskReferenceMode] = None (alias 'maskMode')`

Prompts the model to generate a mask instead of you needing to provide one (unless MASK_MODE_USER_PROVIDED is used).

field segmentation_classes: `Optional[list[int]] = None (alias 'segmentationClasses')`

A list of up to 5 class ids to use for semantic segmentation. Automatically creates an image mask based on specific objects.

class genai.types.**MaskReferenceConfigDict**

Bases: `TypedDict`

Configuration for a Mask reference image.

mask_dilation: `Optional[float]`

Dilation percentage of the mask provided. Float between 0 and 1.

mask_mode: `Optional[MaskReferenceMode]`

Prompts the model to generate a mask instead of you needing to provide one (unless MASK_MODE_USER_PROVIDED is used).

segmentation_classes: `Optional[list[int]]`

A list of up to 5 class ids to use for semantic segmentation. Automatically creates an image mask based on specific objects.

pydantic model genai.types.**MaskReferenceImage**

Bases: `BaseModel`

A mask reference image.

This encapsulates either a mask image provided by the user and configs for the user provided mask, or only config parameters for the model to generate a mask.

A mask image is an image whose non-zero values indicate where to edit the base image. If the user provides a mask image, the mask must be in the same dimensions as the raw image.

Create a new model by parsing and validating input data from keyword arguments.

Skip to content

ıtionError][pydantic_core.ValidationError] if the input data cannot be validated to nodel.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `config (genai.types.MaskReferenceConfig | None)`
- `mask_image_config (genai.types.MaskReferenceConfig | None)`
- `reference_id (int | None)`
- `reference_image (genai.types.Image | None)`
- `reference_type (str | None)`

VALIDATORS:

- `_validate_mask_image_config` » `all fields`

field config: `Optional[MaskReferenceConfig] = None`

Re-map config to mask_reference_config to send to API.

Configuration for the mask reference image.

VALIDATED BY:

- `_validate_mask_image_config`

field mask_image_config: `Optional[MaskReferenceConfig] = None (alias 'maskImageConfig')`

VALIDATED BY:

- `_validate_mask_image_config`

field reference_id: `Optional[int] = None (alias 'referenceId')`

The id of the reference image.

VALIDATED BY:

- `_validate_mask_image_config`

field reference_image: `Optional[Image] = None (alias 'referenceImage')`

The reference image for the editing operation.

VALIDATED BY:

- `_validate_mask_image_config`

field reference_type: `Optional[str] = None (alias 'referenceType')`

The type of the reference image. Only set by the SDK.

VALIDATED BY:

- `_validate_mask_image_config`

class genai.types.**MaskReferenceImageDict**

Bases: `TypedDict`

ence image.

Skip to content

ulates either a mask image provided by the user and configs for the user provided mask, or only config parameters for the model to generate a mask.

A mask image is an image whose non-zero values indicate where to edit the base image. If the user provides a mask image, the mask must be in the same dimensions as the raw image.

config: Optional [**MaskReferenceConfigDict**]

Configuration for the mask reference image.

reference_id: Optional [int]

The id of the reference image.

reference_image: Optional [**ImageDict**]

The reference image for the editing operation.

reference_type: Optional [str]

The type of the reference image. Only set by the SDK.

class genai.types.**MaskReferenceMode**(*values)

Bases: CaseInSensitiveEnum

Enum representing the mask mode of a mask reference image.

MASK_MODE_BACKGROUND = 'MASK_MODE_BACKGROUND'

MASK_MODE_DEFAULT = 'MASK_MODE_DEFAULT'

MASK_MODE_FOREGROUND = 'MASK_MODE_FOREGROUND'

MASK_MODE_SEMANTIC = 'MASK_MODE_SEMANTIC'

MASK_MODE_USER_PROVIDED = 'MASK_MODE_USER_PROVIDED'

class genai.types.**MediaResolution**(*values)

Bases: CaseInSensitiveEnum

The media resolution to use.

MEDIA_RESOLUTION_HIGH = 'MEDIA_RESOLUTION_HIGH'

MEDIA_RESOLUTION_LOW = 'MEDIA_RESOLUTION_LOW'

MEDIA_RESOLUTION_MEDIUM = 'MEDIA_RESOLUTION_MEDIUM'

MEDIA_RESOLUTION_UNSPECIFIED = 'MEDIA_RESOLUTION_UNSPECIFIED'

class genai.types.**Modality**(*values)

Bases: CaseInSensitiveEnum

Server content modalities.

AUDIO = 'AUDIO'

IMAGE = 'IMAGE'

MODALITY_UNSPECIFIED = 'MODALITY_UNSPECIFIED'

Skip to content T'

class genai.types.**Mode**(*values)

Bases: `CaseInSensitiveEnum`

The mode of the predictor to be used in dynamic retrieval.

```
MODE_DYNAMIC = 'MODE_DYNAMIC'
```

```
MODE_UNSPECIFIED = 'MODE_UNSPECIFIED'
```

pydantic model genai.types.**Model**

Skip to content

Bases: `BaseModel`

A trained machine learning model.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `description (str | None)`
- `display_name (str | None)`
- `endpoints (list[genai.types.Endpoint] | None)`
- `input_token_limit (int | None)`
- `labels (dict[str, str] | None)`
- `name (str | None)`
- `output_token_limit (int | None)`
- `supported_actions (list[str] | None)`
- `tuned_model_info (genai.types.TunedModelInfo | None)`
- `version (str | None)`

field description: `Optional[str] = None`

Description of the model.

field display_name: `Optional[str] = None (alias 'displayName')`

Display name of the model.

field endpoints: `Optional[list[Endpoint]] = None`

List of deployed models created from this base model. Note that a model could have been deployed to endpoints in different locations.

field input_token_limit: `Optional[int] = None (alias 'inputTokenLimit')`

The maximum number of input tokens that the model can handle.

field labels: `Optional[dict[str, str]] = None`

Labels with user-defined metadata to organize your models.

field name: `Optional[str] = None`

Resource name of the model.

field output_token_limit: `Optional[int] = None (alias 'outputTokenLimit')`

Skip to content imum number of output tokens that the model can generate.

field supported_actions: `Optional[list[str]] = None (alias 'supportedActions')`

List of actions that are supported by the model.

field tuned_model_info: Optional[TunedModelInfo] = None (alias 'tunedModelInfo')

Information about the tuned model from the base model.

field version: Optional[str] = None

Version ID of the model. A new version is committed when a new model version is uploaded or trained under an existing model ID. The version ID is an auto-incrementing decimal number in string representation.

pydantic model genai.types.ModelContent

Bases: Content

ModelContent facilitates the creation of a Content object with a model role.

Example usages:

- Create a model Content object with a string: model_content = ModelContent("Why is the sky blue?")
- Create a model Content object with a file data Part object: model_content = ModelContent(Part.from_uri(file_uril="gs://bucket/file.txt", mime_type="text/plain"))
- Create a model Content object with byte data Part object: model_content = ModelContent(Part.from_bytes(data=b"Hello, World!", mime_type="text/plain"))

  You can create a model Content object using other classmethods in the Part class as well. You can also create a model Content using a list of Part objects or strings.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- parts (list[genai.types.Part])
- role (Literal['model'])

field parts: list[Part] [Required]

field role: Literal['model'] = 'model'

class genai.types.ModelDict

Skip to content

Bases: `TypedDict`

A trained machine learning model.

description: `Optional[str]`

Description of the model.

display_name: `Optional[str]`

Display name of the model.

endpoints: `Optional[list[EndpointDict]]`

List of deployed models created from this base model. Note that a model could have been deployed to endpoints in different locations.

input_token_limit: `Optional[int]`

The maximum number of input tokens that the model can handle.

labels: `Optional[dict[str, str]]`

Labels with user-defined metadata to organize your models.

name: `Optional[str]`

Resource name of the model.

output_token_limit: `Optional[int]`

The maximum number of output tokens that the model can generate.

supported_actions: `Optional[list[str]]`

List of actions that are supported by the model.

tuned_model_info: `Optional[TunedModelInfoDict]`

Information about the tuned model from the base model.

version: `Optional[str]`

Version ID of the model. A new version is committed when a new model version is uploaded or trained under an existing model ID. The version ID is an auto-incrementing decimal number in string representation.

pydantic model genai.types.**Operation**

Bases: `BaseModel`

A long-running operation.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

Skip to content ly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `done (bool | None)`
- `error (dict[str, Any] | None)`
- `metadata (dict[str, Any] | None)`
- `name (str | None)`
- `response (dict[str, Any] | None)`

field done: `Optional[bool] = None`

If the value is *false*, it means the operation is still in progress. If *true*, the operation is completed, and either *error* or *response* is available.

field error: `Optional[dict[str, Any]] = None`

The error result of the operation in case of failure or cancellation.

field metadata: `Optional[dict[str, Any]] = None`

Service-specific metadata associated with the operation. It typically contains progress information and common metadata such as create time. Some services might not provide such metadata. Any method that returns a long-running operation should document the metadata type, if any.

field name: `Optional[str] = None`

The server-assigned name, which is only unique within the same service that originally returns it. If you use the default HTTP mapping, the *name* should be a resource name ending with *operations/{unique_id}*.

field response: `Optional[dict[str, Any]] = None`

The normal response of the operation in case of success.

class genai.types.**OperationDict**

Skip to content

Bases: `TypedDict`

A long-running operation.

done: `Optional[bool]`

If the value is *false*, it means the operation is still in progress. If *true*, the operation is completed, and either *error* or *response* is available.

error: `Optional[dict[str, Any]]`

The error result of the operation in case of failure or cancellation.

metadata: `Optional[dict[str, Any]]`

Service-specific metadata associated with the operation. It typically contains progress information and common metadata such as create time. Some services might not provide such metadata. Any method that returns a long-running operation should document the metadata type, if any.

name: `Optional[str]`

The server-assigned name, which is only unique within the same service that originally returns it. If you use the default HTTP mapping, the *name* should be a resource name ending with *operations/{unique_id}*.

response: `Optional[dict[str, Any]]`

The normal response of the operation in case of success.

class genai.types.**Outcome**(*values)

Bases: `CaseInSensitiveEnum`

Required. Outcome of the code execution.

OUTCOME_DEADLINE_EXCEEDED = 'OUTCOME_DEADLINE_EXCEEDED'

OUTCOME_FAILED = 'OUTCOME_FAILED'

OUTCOME_OK = 'OUTCOME_OK'

OUTCOME_UNSPECIFIED = 'OUTCOME_UNSPECIFIED'

pydantic model genai.types.**Part**

Bases: `BaseModel`

A datatype containing media content.

Exactly one field within a Part should be set, representing the specific type of content being conveyed. Using multiple fields within the same *Part* instance is considered invalid.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to

Skip to content model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `code_execution_result (genai.types.CodeExecutionResult | None)`
- `executable_code (genai.types.ExecutableCode | None)`
- `file_data (genai.types.FileData | None)`
- `function_call (genai.types.FunctionCall | None)`
- `function_response (genai.types.FunctionResponse | None)`
- `inline_data (genai.types.Blob | None)`
- `text (str | None)`
- `thought (bool | None)`
- `video_metadata (genai.types.VideoMetadata | None)`

field code_execution_result: `Optional[CodeExecutionResult] = None (alias 'codeExecutionResult')`

Optional. Result of executing the [ExecutableCode].

field executable_code: `Optional[ExecutableCode] = None (alias 'executableCode')`

Optional. Code generated by the model that is meant to be executed.

field file_data: `Optional[FileData] = None (alias 'fileData')`

Optional. URI based data.

field function_call: `Optional[FunctionCall] = None (alias 'functionCall')`

Optional. A predicted [FunctionCall] returned from the model that contains a string representing the [FunctionDeclaration.name] with the parameters and their values.

field function_response: `Optional[FunctionResponse] = None (alias 'functionResponse')`

Optional. The result output of a [FunctionCall] that contains a string representing the [FunctionDeclaration.name] and a structured JSON object containing any output from the function call. It is used as context to the model.

field inline_data: `Optional[Blob] = None (alias 'inlineData')`

Optional. Inlined bytes data.

field text: `Optional[str] = None`

Optional. Text part (can be code).

field thought: `Optional[bool] = None`

Indicates if the part is thought from the model.

field video_metadata: `Optional[VideoMetadata] = None (alias 'videoMetadata')`

`a for a given video.

Skip to content

classmethod from_bytes(*, data, mime_type)

RETURN TYPE:

Part

classmethod from_code_execution_result(*, outcome, output)

RETURN TYPE:

Part

classmethod from_executable_code(*, code, language)

RETURN TYPE:

Part

classmethod from_function_call(*, name, args)

RETURN TYPE:

Part

classmethod from_function_response(*, name, response)

RETURN TYPE:

Part

classmethod from_text(*, text)

RETURN TYPE:

Part

classmethod from_uri(*, file_uri, mime_type)

RETURN TYPE:

Part

classmethod from_video_metadata(*, start_offset, end_offset)

RETURN TYPE:

Part

class genai.types.PartDict

Skip to content

Bases: `TypedDict`

A datatype containing media content.

Exactly one field within a Part should be set, representing the specific type of content being conveyed. Using multiple fields within the same *Part* instance is considered invalid.

code_execution_result: `Optional[CodeExecutionResultDict]`

Optional. Result of executing the [ExecutableCode].

executable_code: `Optional[ExecutableCodeDict]`

Optional. Code generated by the model that is meant to be executed.

file_data: `Optional[FileDataDict]`

Optional. URI based data.

function_call: `Optional[FunctionCallDict]`

Optional. A predicted [FunctionCall] returned from the model that contains a string representing the [FunctionDeclaration.name] with the parameters and their values.

function_response: `Optional[FunctionResponseDict]`

Optional. The result output of a [FunctionCall] that contains a string representing the [FunctionDeclaration.name] and a structured JSON object containing any output from the function call. It is used as context to the model.

inline_data: `Optional[BlobDict]`

Optional. Inlined bytes data.

text: `Optional[str]`

Optional. Text part (can be code).

thought: `Optional[bool]`

Indicates if the part is thought from the model.

video_metadata: `Optional[VideoMetadataDict]`

Metadata for a given video.

pydantic model genai.types.**PartnerModelTuningSpec**

Bases: `BaseModel`

Tuning spec for Partner models.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

Skip to content

:ly positional-only to allow *self* as a field name.

► Show JSON schema

FIELDS:

- `hyper_parameters (dict[str, Any] | None)`
- `training_dataset_uri (str | None)`
- `validation_dataset_uri (str | None)`

field hyper_parameters: `Optional[dict[str, Any]] = None (alias 'hyperParameters')`

Hyperparameters for tuning. The accepted hyper_parameters and their valid range of values will differ depending on the base model.

field training_dataset_uri: `Optional[str] = None (alias 'trainingDatasetUri')`

Required. Cloud Storage path to file containing training dataset for tuning. The dataset must be formatted as a JSONL file.

field validation_dataset_uri: `Optional[str] = None (alias 'validationDatasetUri')`

Optional. Cloud Storage path to file containing validation dataset for tuning. The dataset must be formatted as a JSONL file.

class genai.types.**PartnerModelTuningSpecDict**

Bases: `TypedDict`

Tuning spec for Partner models.

hyper_parameters: `Optional[dict[str, Any]]`

Hyperparameters for tuning. The accepted hyper_parameters and their valid range of values will differ depending on the base model.

training_dataset_uri: `Optional[str]`

Required. Cloud Storage path to file containing training dataset for tuning. The dataset must be formatted as a JSONL file.

validation_dataset_uri: `Optional[str]`

Optional. Cloud Storage path to file containing validation dataset for tuning. The dataset must be formatted as a JSONL file.

class genai.types.**PersonGeneration**(*values)

Bases: `CaseInSensitiveEnum`

Enum that controls the generation of people.

ALLOW_ADULT = 'ALLOW_ADULT'

ALLOW_ALL = 'ALLOW_ALL'

DONT_ALLOW = 'DONT_ALLOW'

pydantic model genai.types.**PrebuiltVoiceConfig**

Skip to content `odel`

The configuration for the prebuilt speaker to use.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `voice_name (str | None)`

field voice_name: `Optional[str] = None (alias 'voiceName')`

The name of the prebuilt voice to use.

class genai.types.**PrebuiltVoiceConfigDict**

Bases: `TypedDict`

The configuration for the prebuilt speaker to use.

voice_name: `Optional[str]`

The name of the prebuilt voice to use.

pydantic model genai.types.**RawReferenceImage**

Skip to content

Bases: `BaseModel`

A raw reference image.

A raw reference image represents the base image to edit, provided by the user. It can optionally be provided in addition to a mask reference image or a style reference image.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `reference_id (int | None)`
- `reference_image (genai.types.Image | None)`
- `reference_type (str | None)`

VALIDATORS:

- `_validate_mask_image_config` » `all fields`

field reference_id: `Optional[int] = None (alias 'referenceId')`

The id of the reference image.

VALIDATED BY:

- `_validate_mask_image_config`

field reference_image: `Optional[Image] = None (alias 'referenceImage')`

The reference image for the editing operation.

VALIDATED BY:

- `_validate_mask_image_config`

field reference_type: `Optional[str] = None (alias 'referenceType')`

The type of the reference image. Only set by the SDK.

VALIDATED BY:

- `_validate_mask_image_config`

class genai.types.**RawReferenceImageDict**

Skip to content

Bases: `TypedDict`

A raw reference image.

A raw reference image represents the base image to edit, provided by the user. It can optionally be provided in addition to a mask reference image or a style reference image.

reference_id: `Optional[int]`

The id of the reference image.

reference_image: `Optional[ImageDict]`

The reference image for the editing operation.

reference_type: `Optional[str]`

The type of the reference image. Only set by the SDK.

pydantic model genai.types.ReplayFile

Bases: `BaseModel`

Represents a recorded session.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `interactions (list[genai.types.ReplayInteraction] | None)`
- `replay_id (str | None)`

field interactions: `Optional[list[ReplayInteraction]] = None`

field replay_id: `Optional[str] = None (alias 'replayId')`

class genai.types.ReplayFileDict

Bases: `TypedDict`

Represents a recorded session.

interactions: `Optional[list[ReplayInteractionDict]]`

replay_id: `Optional[str]`

pydantic model genai.types.ReplayInteraction

Bases: `BaseModel`

Skip to content single interaction, request and response in a replay.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `request (genai.types.ReplayRequest | None)`
- `response (genai.types.ReplayResponse | None)`

field **request**: `Optional[ReplayRequest] = None`

field **response**: `Optional[ReplayResponse] = None`

class genai.types.**ReplayInteractionDict**

Bases: `TypedDict`

Represents a single interaction, request and response in a replay.

request: `Optional[ReplayRequestDict]`

response: `Optional[ReplayResponseDict]`

pydantic model genai.types.**ReplayRequest**

Bases: `BaseModel`

Represents a single request in a replay.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `body_segments (list[dict[str, Any]] | None)`
- `headers (dict[str, str] | None)`
- `method (str | None)`
- `url (str | None)`

field **body_segments**: `Optional[list[dict[str, Any]]] = None (alias 'bodySegments')`

field **headers**: `Optional[dict[str, str]] = None`

field **method**: `Optional[str] = None`

field **url**: `Optional[str] = None`

Skip to content ...es.**ReplayRequestDict**

Bases: `TypedDict`

Represents a single request in a replay.

body_segments: `Optional[list[dict[str, Any]]]`

headers: `Optional[dict[str, str]]`

method: `Optional[str]`

url: `Optional[str]`

pydantic model genai.types.**ReplayResponse**

Bases: `BaseModel`

Represents a single response in a replay.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `body_segments (list[dict[str, Any]] | None)`
- `headers (dict[str, str] | None)`
- `sdk_response_segments (list[dict[str, Any]] | None)`
- `status_code (int | None)`

field body_segments: `Optional[list[dict[str, Any]]]` **= None (alias 'bodySegments')**

field headers: `Optional[dict[str, str]]` **= None**

field sdk_response_segments: `Optional[list[dict[str, Any]]]` **= None (alias 'sdkResponseSegments')**

field status_code: `Optional[int]` **= None (alias 'statusCode')**

class genai.types.**ReplayResponseDict**

Bases: `TypedDict`

Represents a single response in a replay.

body_segments: `Optional[list[dict[str, Any]]]`

headers: `Optional[dict[str, str]]`

sdk_response_segments: `Optional[list[dict[str, Any]]]`

status_code: `Optional[int]`

pydantic model genai.types.**Retrieval**

Bases: `odel`

Skip to content

ieval tool that model can call to access external knowledge.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `disable_attribution (bool | None)`
- `vertex_ai_search (genai.types.VertexAISearch | None)`
- `vertex_rag_store (genai.types.VertexRagStore | None)`

field disable_attribution: `Optional[bool] = None (alias 'disableAttribution')`

Optional. Deprecated. This option is no longer supported.

field vertex_ai_search: `Optional[VertexAISearch] = None (alias 'vertexAiSearch')`

Set to use data source powered by Vertex AI Search.

field vertex_rag_store: `Optional[VertexRagStore] = None (alias 'vertexRagStore')`

Set to use data source powered by Vertex RAG store. User data is uploaded via the VertexRagDataService.

class genai.types.**RetrievalDict**

Bases: `TypedDict`

Defines a retrieval tool that model can call to access external knowledge.

disable_attribution: `Optional[bool]`

Optional. Deprecated. This option is no longer supported.

vertex_ai_search: `Optional[VertexAISearchDict]`

Set to use data source powered by Vertex AI Search.

vertex_rag_store: `Optional[VertexRagStoreDict]`

Set to use data source powered by Vertex RAG store. User data is uploaded via the VertexRagDataService.

pydantic model genai.types.**RetrievalMetadata**

Bases: `BaseModel`

Metadata related to retrieval in the grounding flow.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

Skip to content

ly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `google_search_dynamic_retrieval_score (float | None)`

field **google_search_dynamic_retrieval_score**: `Optional[float] = None (alias 'googleSearchDynamicRetrievalScore')`

Optional. Score indicating how likely information from Google Search could help answer the prompt. The score is in the range *[0, 1]*, where 0 is the least likely and 1 is the most likely. This score is only populated when Google Search grounding and dynamic retrieval is enabled. It will be compared to the threshold to determine whether to trigger Google Search.

class genai.types.**RetrievalMetadataDict**

Bases: `TypedDict`

Metadata related to retrieval in the grounding flow.

google_search_dynamic_retrieval_score: `Optional[float]`

Optional. Score indicating how likely information from Google Search could help answer the prompt. The score is in the range *[0, 1]*, where 0 is the least likely and 1 is the most likely. This score is only populated when Google Search grounding and dynamic retrieval is enabled. It will be compared to the threshold to determine whether to trigger Google Search.

pydantic model genai.types.**SafetyAttributes**

Bases: `BaseModel`

Safety attributes of a GeneratedImage or the user-provided prompt.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `categories (list[str] | None)`
- `scores (list[float] | None)`

field **categories**: `Optional[list[str]] = None`

List of RAI categories.

field **scores**: `Optional[list[float]] = None`

List of scores of each categories.

class genai.types.**SafetyAttributesDict**

`Dict`

Skip to content

Safety attributes of a GeneratedImage or the user-provided prompt.

categories: `Optional[list[str]]`

List of RAI categories.

scores: `Optional[list[float]]`

List of scores of each categories.

class genai.types.**SafetyFilterLevel**(*values)

Bases: `CaseInSensitiveEnum`

Enum that controls the safety filter level for objectionable content.

BLOCK_LOW_AND_ABOVE = 'BLOCK_LOW_AND_ABOVE'

BLOCK_MEDIUM_AND_ABOVE = 'BLOCK_MEDIUM_AND_ABOVE'

BLOCK_NONE = 'BLOCK_NONE'

BLOCK_ONLY_HIGH = 'BLOCK_ONLY_HIGH'

pydantic model genai.types.**SafetyRating**

Skip to content

Bases: `BaseModel`

Safety rating corresponding to the generated content.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `blocked (bool | None)`
- `category (genai.types.HarmCategory | None)`
- `probability (genai.types.HarmProbability | None)`
- `probability_score (float | None)`
- `severity (genai.types.HarmSeverity | None)`
- `severity_score (float | None)`

field blocked: `Optional[bool] = None`

Output only. Indicates whether the content was filtered out because of this rating.

field category: `Optional[HarmCategory] = None`

Output only. Harm category.

field probability: `Optional[HarmProbability] = None`

Output only. Harm probability levels in the content.

field probability_score: `Optional[float] = None (alias 'probabilityScore')`

Output only. Harm probability score.

field severity: `Optional[HarmSeverity] = None`

Output only. Harm severity levels in the content.

field severity_score: `Optional[float] = None (alias 'severityScore')`

Output only. Harm severity score.

class genai.types.**SafetyRatingDict**

Skip to content

Bases: `TypedDict`

Safety rating corresponding to the generated content.

blocked: `Optional[bool]`

Output only. Indicates whether the content was filtered out because of this rating.

category: `Optional[HarmCategory]`

Output only. Harm category.

probability: `Optional[HarmProbability]`

Output only. Harm probability levels in the content.

probability_score: `Optional[float]`

Output only. Harm probability score.

severity: `Optional[HarmSeverity]`

Output only. Harm severity levels in the content.

severity_score: `Optional[float]`

Output only. Harm severity score.

pydantic model genai.types.SafetySetting

Bases: `BaseModel`

Safety settings.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `category (genai.types.HarmCategory | None)`
- `method (genai.types.HarmBlockMethod | None)`
- `threshold (genai.types.HarmBlockThreshold | None)`

field category: `Optional[HarmCategory] = None`

Required. Harm category.

field method: `Optional[HarmBlockMethod] = None`

Determines if the harm block method uses probability or probability and severity scores.

hold: `Optional[HarmBlockThreshold] = None`

. The harm block threshold.

Skip to content

class genai.types.SafetySettingDict

Bases: `TypedDict`

Safety settings.

category: `Optional[HarmCategory]`

Required. Harm category.

method: `Optional[HarmBlockMethod]`

Determines if the harm block method uses probability or probability and severity scores.

threshold: `Optional[HarmBlockThreshold]`

Required. The harm block threshold.

pydantic model `genai.types.Schema`

Bases: `BaseModel`

Schema that defines the format of input and output data.

Represents a select subset of an OpenAPI 3.0 schema object.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

Skip to content

FIELDS:

- `any_of (list[genai.types.Schema] | None)`
- `default (Any | None)`
- `description (str | None)`
- `enum (list[str] | None)`
- `example (Any | None)`
- `format (str | None)`
- `items (genai.types.Schema | None)`
- `max_items (int | None)`
- `max_length (int | None)`
- `max_properties (int | None)`
- `maximum (float | None)`
- `min_items (int | None)`
- `min_length (int | None)`
- `min_properties (int | None)`
- `minimum (float | None)`
- `nullable (bool | None)`
- `pattern (str | None)`
- `properties (dict[str, genai.types.Schema] | None)`
- `property_ordering (list[str] | None)`
- `required (list[str] | None)`
- `title (str | None)`
- `type (genai.types.Type | None)`

field any_of: `Optional[list[Schema]] = None (alias 'anyOf')`

Optional. The value should be validated against any (one or more) of the subschemas in the list.

field default: `Optional[Any] = None`

Optional. Default value of the data.

field description: `Optional[str] = None`

Optional. The description of the data.

field enum: `Optional[list[str]] = None`

Optional. Possible values of the element of primitive type with enum format. Examples: 1. We can define direction as : {type:STRING, format:enum, enum:["EAST", NORTH", "SOUTH", "WEST"]} 2. We can define apartment number as : {type:INTEGER, format:enum, enum:["101", "201", "301"]}

Skip to content **le:** `Optional[Any] = None`

Optional. Example of the object. Will only populated when the object is the root.

field format: `Optional[str] = None`

Optional. The format of the data. Supported formats: for NUMBER type: “float”, “double” for INTEGER type: “int32”, “int64” for STRING type: “email”, “byte”, etc

field items: `Optional[Schema] = None`

Optional. SCHEMA FIELDS FOR TYPE ARRAY Schema of the elements of Type.ARRAY.

field max_items: `Optional[int] = None (alias 'maxItems')`

Optional. Maximum number of the elements for Type.ARRAY.

field max_length: `Optional[int] = None (alias 'maxLength')`

Optional. Maximum length of the Type.STRING

field max_properties: `Optional[int] = None (alias 'maxProperties')`

Optional. Maximum number of the properties for Type.OBJECT.

field maximum: `Optional[float] = None`

Optional. Maximum value of the Type.INTEGER and Type.NUMBER

field min_items: `Optional[int] = None (alias 'minItems')`

Optional. Minimum number of the elements for Type.ARRAY.

field min_length: `Optional[int] = None (alias 'minLength')`

Optional. SCHEMA FIELDS FOR TYPE STRING Minimum length of the Type.STRING

field min_properties: `Optional[int] = None (alias 'minProperties')`

Optional. Minimum number of the properties for Type.OBJECT.

field minimum: `Optional[float] = None`

Optional. SCHEMA FIELDS FOR TYPE INTEGER and NUMBER Minimum value of the Type.INTEGER and Type.NUMBER

field nullable: `Optional[bool] = None`

Optional. Indicates if the value may be null.

field pattern: `Optional[str] = None`

Optional. Pattern of the Type.STRING to restrict a string to a regular expression.

field properties: `Optional[dict[str, Schema]] = None`

Optional. SCHEMA FIELDS FOR TYPE OBJECT Properties of Type.OBJECT.

field property_ordering: `Optional[list[str]] = None (alias 'propertyOrdering')`

Optional. The order of the properties. Not a standard field in open api spec. Only used to support the order of the properties.

Skip to content

red: `Optional[list[str]] = None`

Optional. Required properties of Type.OBJECT.

field title: `Optional[str] = None`

Optional. The title of the Schema.

field type: `Optional[Type] = None`

Optional. The type of the data.

class genai.types.**SchemaDict**

Skip to content

Bases: `TypedDict`

Schema that defines the format of input and output data.

Represents a select subset of an OpenAPI 3.0 schema object.

any_of: `Optional[list[SchemaDict]]`

Optional. The value should be validated against any (one or more) of the subschemas in the list.

default: `Optional[Any]`

Optional. Default value of the data.

description: `Optional[str]`

Optional. The description of the data.

enum: `Optional[list[str]]`

{type:STRING, format:enum, enum:["EAST", NORTH", "SOUTH", "WEST"]} 2. We can define apartment number as : {type:INTEGER, format:enum, enum:["101", "201", "301"]}

TYPE:

Optional. Possible values of the element of primitive type with enum format. Examples

TYPE:

1. We can define direction as

example: `Optional[Any]`

Optional. Example of the object. Will only populated when the object is the root.

format: `Optional[str]`

"float", "double" for INTEGER type: "int32", "int64" for STRING type: "email", "byte", etc

TYPE:

Optional. The format of the data. Supported formats

TYPE:

for NUMBER type

max_items: `Optional[int]`

Optional. Maximum number of the elements for Type.ARRAY.

max_length: `Optional[int]`

Optional. Maximum length of the Type.STRING

max_properties: `Optional[int]`

Optional. Maximum number of the properties for Type.OBJECT.

`tional[float]`

Skip to content

Maximum value of the Type.INTEGER and Type.NUMBER

min_items: `Optional[int]`

Optional. Minimum number of the elements for Type.ARRAY.

min_length: Optional[int]

Optional. SCHEMA FIELDS FOR TYPE STRING Minimum length of the Type.STRING

min_properties: Optional[int]

Optional. Minimum number of the properties for Type.OBJECT.

minimum: Optional[float]

Optional. SCHEMA FIELDS FOR TYPE INTEGER and NUMBER Minimum value of the Type.INTEGER and Type.NUMBER

nullable: Optional[bool]

Optional. Indicates if the value may be null.

pattern: Optional[str]

Optional. Pattern of the Type.STRING to restrict a string to a regular expression.

properties: Optional[dict[str, SchemaDict]]

Optional. SCHEMA FIELDS FOR TYPE OBJECT Properties of Type.OBJECT.

property_ordering: Optional[list[str]]

Optional. The order of the properties. Not a standard field in open api spec. Only used to support the order of the properties.

required: Optional[list[str]]

Optional. Required properties of Type.OBJECT.

title: Optional[str]

Optional. The title of the Schema.

type: Optional[Type]

Optional. The type of the data.

pydantic model genai.types.**SearchEntryPoint**

Bases: BaseModel

Google search entry point.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

Skip to content

FIELDS:

- `rendered_content (str | None)`
- `sdk_blob (bytes | None)`

field rendered_content: `Optional[str] = None (alias 'renderedContent')`

Optional. Web content snippet that can be embedded in a web page or an app webview.

field sdk_blob: `Optional[bytes] = None (alias 'sdkBlob')`

Optional. Base64 encoded JSON representing array of tuple.

class genai.types.SearchEntryPointDict

Bases: `TypedDict`

Google search entry point.

rendered_content: `Optional[str]`

Optional. Web content snippet that can be embedded in a web page or an app webview.

sdk_blob: `Optional[bytes]`

Optional. Base64 encoded JSON representing array of tuple.

pydantic model genai.types.Segment

Skip to content

Bases: `BaseModel`

Segment of the content.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `end_index (int | None)`
- `part_index (int | None)`
- `start_index (int | None)`
- `text (str | None)`

field **end_index**: `Optional[int] = None (alias 'endIndex')`

Output only. End index in the given Part, measured in bytes. Offset from the start of the Part, exclusive, starting at zero.

field **part_index**: `Optional[int] = None (alias 'partIndex')`

Output only. The index of a Part object within its parent Content object.

field **start_index**: `Optional[int] = None (alias 'startIndex')`

Output only. Start index in the given Part, measured in bytes. Offset from the start of the Part, inclusive, starting at zero.

field **text**: `Optional[str] = None`

Output only. The text corresponding to the segment from the response.

class genai.types.**SegmentDict**

Skip to content

Bases: `TypedDict`

Segment of the content.

end_index: `Optional[int]`

Output only. End index in the given Part, measured in bytes. Offset from the start of the Part, exclusive, starting at zero.

part_index: `Optional[int]`

Output only. The index of a Part object within its parent Content object.

start_index: `Optional[int]`

Output only. Start index in the given Part, measured in bytes. Offset from the start of the Part, inclusive, starting at zero.

text: `Optional[str]`

Output only. The text corresponding to the segment from the response.

pydantic model genai.types.**SpeechConfig**

Bases: `BaseModel`

The speech generation configuration.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `voice_config (genai.types.VoiceConfig | None)`

field voice_config: `Optional[VoiceConfig] = None (alias 'voiceConfig')`

The configuration for the speaker to use.

class genai.types.**SpeechConfigDict**

Bases: `TypedDict`

The speech generation configuration.

voice_config: `Optional[VoiceConfigDict]`

The configuration for the speaker to use.

class genai.types.**State**(*values)

Skip to content

Bases: `CaseInSensitiveEnum`

Output only. RagFile state.

```
ACTIVE = 'ACTIVE'
ERROR = 'ERROR'
STATE_UNSPECIFIED = 'STATE_UNSPECIFIED'
```

pydantic model genai.types.**StyleReferenceConfig**

Bases: `BaseModel`

Configuration for a Style reference image.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `style_description (str | None)`

field style_description: `Optional[str] = None (alias 'styleDescription')`

A text description of the style to use for the generated image.

class genai.types.**StyleReferenceConfigDict**

Bases: `TypedDict`

Configuration for a Style reference image.

style_description: `Optional[str]`

A text description of the style to use for the generated image.

pydantic model genai.types.**StyleReferenceImage**

Bases: `BaseModel`

A style reference image.

This encapsulates a style reference image provided by the user, and additionally optional config parameters for the style reference image.

A raw reference image can also be provided as a destination for the style to be applied to.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `config (genai.types.StyleReferenceConfig | None)`
- `reference_id (int | None)`
- `reference_image (genai.types.Image | None)`
- `reference_type (str | None)`
- `style_image_config (genai.types.StyleReferenceConfig | None)`

VALIDATORS:

- `_validate_mask_image_config` » `all fields`

field config: `Optional[StyleReferenceConfig] = None`

Re-map config to style_reference_config to send to API.

Configuration for the style reference image.

VALIDATED BY:

- `_validate_mask_image_config`

field reference_id: `Optional[int] = None (alias 'referenceId')`

The id of the reference image.

VALIDATED BY:

- `_validate_mask_image_config`

field reference_image: `Optional[Image] = None (alias 'referenceImage')`

The reference image for the editing operation.

VALIDATED BY:

- `_validate_mask_image_config`

field reference_type: `Optional[str] = None (alias 'referenceType')`

The type of the reference image. Only set by the SDK.

VALIDATED BY:

- `_validate_mask_image_config`

field style_image_config: `Optional[StyleReferenceConfig] = None (alias 'styleImageConfig')`

VALIDATED BY:

- `_validate_mask_image_config`

class genai.types.**StyleReferenceImageDict**

Bases: `TypedDict`

nce image.

Skip to content

ulates a style reference image provided by the user, and additionally optional config parameters for the style reference image.

A raw reference image can also be provided as a destination for the style to be applied to.

config: Optional[StyleReferenceConfigDict]

Configuration for the style reference image.

reference_id: Optional[int]

The id of the reference image.

reference_image: Optional[ImageDict]

The reference image for the editing operation.

reference_type: Optional[str]

The type of the reference image. Only set by the SDK.

pydantic model genai.types.**SubjectReferenceConfig**

Bases: BaseModel

Configuration for a Subject reference image.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- subject_description (str | None)
- subject_type (genai.types.SubjectReferenceType | None)

field subject_description: Optional[str] = None (alias 'subjectDescription')

Subject description for the image.

field subject_type: Optional[SubjectReferenceType] = None (alias 'subjectType')

The subject type of a subject reference image.

class genai.types.**SubjectReferenceConfigDict**

Bases: TypedDict

Configuration for a Subject reference image.

subject_description: Optional[str]

Subject description for the image.

subject_type: Optional[SubjectReferenceType]

ect type of a subject reference image.

Skip to content

pydantic model genai.types.**SubjectReferenceImage**

Bases: BaseModel

A subject reference image.

This encapsulates a subject reference image provided by the user, and additionally optional config parameters for the subject reference image.

A raw reference image can also be provided as a destination for the subject to be applied to.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `config (genai.types.SubjectReferenceConfig | None)`
- `reference_id (int | None)`
- `reference_image (genai.types.Image | None)`
- `reference_type (str | None)`
- `subject_image_config (genai.types.SubjectReferenceConfig | None)`

VALIDATORS:

- `_validate_mask_image_config` » `all fields`

field config: `Optional[SubjectReferenceConfig] = None`

Re-map config to subject_reference_config to send to API.

Configuration for the subject reference image.

VALIDATED BY:

- `_validate_mask_image_config`

field reference_id: `Optional[int] = None (alias 'referenceId')`

The id of the reference image.

VALIDATED BY:

- `_validate_mask_image_config`

field reference_image: `Optional[Image] = None (alias 'referenceImage')`

The reference image for the editing operation.

VALIDATED BY:

- `_validate_mask_image_config`

field reference_type: `Optional[str] = None (alias 'referenceType')`

The type of the reference image. Only set by the SDK.

Skip to content D BY:

- `_validate_mask_image_config`

field `subject_image_config`: `Optional`[`SubjectReferenceConfig`] = `None` (alias `'subjectImageConfig'`)

VALIDATED BY:

- `_validate_mask_image_config`

class genai.types.**SubjectReferenceImageDict**

Bases: `TypedDict`

A subject reference image.

This encapsulates a subject reference image provided by the user, and additionally optional config parameters for the subject reference image.

A raw reference image can also be provided as a destination for the subject to be applied to.

config: `Optional`[**SubjectReferenceConfigDict**]

Configuration for the subject reference image.

reference_id: `Optional`[`int`]

The id of the reference image.

reference_image: `Optional`[**ImageDict**]

The reference image for the editing operation.

reference_type: `Optional`[`str`]

The type of the reference image. Only set by the SDK.

class genai.types.**SubjectReferenceType**(*values)

Bases: `CaseInSensitiveEnum`

Enum representing the subject type of a subject reference image.

SUBJECT_TYPE_ANIMAL = 'SUBJECT_TYPE_ANIMAL'

SUBJECT_TYPE_DEFAULT = 'SUBJECT_TYPE_DEFAULT'

SUBJECT_TYPE_PERSON = 'SUBJECT_TYPE_PERSON'

SUBJECT_TYPE_PRODUCT = 'SUBJECT_TYPE_PRODUCT'

pydantic model genai.types.**SupervisedHyperParameters**

Bases: `BaseModel`

Hyperparameters for SFT.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

Skip to content

ly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `adapter_size (genai.types.AdapterSize | None)`
- `epoch_count (int | None)`
- `learning_rate_multiplier (float | None)`

field adapter_size: `Optional[AdapterSize] = None (alias 'adapterSize')`

Optional. Adapter size for tuning.

field epoch_count: `Optional[int] = None (alias 'epochCount')`

Optional. Number of complete passes the model makes over the entire training dataset during training.

field learning_rate_multiplier: `Optional[float] = None (alias 'learningRateMultiplier')`

Optional. Multiplier for adjusting the default learning rate.

class genai.types.**SupervisedHyperParametersDict**

Bases: `TypedDict`

Hyperparameters for SFT.

adapter_size: `Optional[AdapterSize]`

Optional. Adapter size for tuning.

epoch_count: `Optional[int]`

Optional. Number of complete passes the model makes over the entire training dataset during training.

learning_rate_multiplier: `Optional[float]`

Optional. Multiplier for adjusting the default learning rate.

pydantic model genai.types.**SupervisedTuningDataStats**

Bases: `BaseModel`

Tuning data statistics for Supervised Tuning.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

Skip to content

FIELDS:

- `total_billable_character_count (int | None)`
- `total_billable_token_count (int | None)`
- `total_truncated_example_count (int | None)`
- `total_tuning_character_count (int | None)`
- `truncated_example_indices (list[int] | None)`
- `tuning_dataset_example_count (int | None)`
- `tuning_step_count (int | None)`
- `user_dataset_examples (list[genai.types.Content] | None)`
- `user_input_token_distribution (genai.types.SupervisedTuningDatasetDistribution | None)`
- `user_message_per_example_distribution (genai.types.SupervisedTuningDatasetDistribution | None)`
- `user_output_token_distribution (genai.types.SupervisedTuningDatasetDistribution | None)`

field total_billable_character_count: Optional[int] = None (alias 'totalBillableCharacterCount')

Output only. Number of billable characters in the tuning dataset.

field total_billable_token_count: Optional[int] = None (alias 'totalBillableTokenCount')

Output only. Number of billable tokens in the tuning dataset.

field total_truncated_example_count: Optional[int] = None (alias 'totalTruncatedExampleCount')

The number of examples in the dataset that have been truncated by any amount.

field total_tuning_character_count: Optional[int] = None (alias 'totalTuningCharacterCount')

Output only. Number of tuning characters in the tuning dataset.

field truncated_example_indices: Optional[list[int]] = None (alias 'truncatedExampleIndices')

A partial sample of the indices (starting from 1) of the truncated examples.

field tuning_dataset_example_count: Optional[int] = None (alias 'tuningDatasetExampleCount')

Output only. Number of examples in the tuning dataset.

field tuning_step_count: Optional[int] = None (alias 'tuningStepCount')

Output only. Number of tuning steps for this Tuning Job.

Skip to content

dataset_examples: Optional[list[Content]] = None (alias ıtasetExamples')

Output only. Sample user messages in the training dataset uri.

field user_input_token_distribution: Optional[SupervisedTuningDatasetDistribution] = None (alias 'userInputTokenDistribution')

Output only. Dataset distributions for the user input tokens.

field user_message_per_example_distribution: Optional[SupervisedTuningDatasetDistribution] = None (alias 'userMessagePerExampleDistribution')

Output only. Dataset distributions for the messages per example.

field user_output_token_distribution: Optional[SupervisedTuningDatasetDistribution] = None (alias 'userOutputTokenDistribution')

Output only. Dataset distributions for the user output tokens.

class genai.types.SupervisedTuningDataStatsDict

Skip to content

Bases: `TypedDict`

Tuning data statistics for Supervised Tuning.

total_billable_character_count: `Optional[int]`

Output only. Number of billable characters in the tuning dataset.

total_billable_token_count: `Optional[int]`

Output only. Number of billable tokens in the tuning dataset.

total_truncated_example_count: `Optional[int]`

The number of examples in the dataset that have been truncated by any amount.

total_tuning_character_count: `Optional[int]`

Output only. Number of tuning characters in the tuning dataset.

truncated_example_indices: `Optional[list[int]]`

A partial sample of the indices (starting from 1) of the truncated examples.

tuning_dataset_example_count: `Optional[int]`

Output only. Number of examples in the tuning dataset.

tuning_step_count: `Optional[int]`

Output only. Number of tuning steps for this Tuning Job.

user_dataset_examples: `Optional[list[ContentDict]]`

Output only. Sample user messages in the training dataset uri.

user_input_token_distribution: `Optional[SupervisedTuningDatasetDistributionDict]`

Output only. Dataset distributions for the user input tokens.

user_message_per_example_distribution: `Optional[SupervisedTuningDatasetDistributionDict]`

Output only. Dataset distributions for the messages per example.

user_output_token_distribution: `Optional[SupervisedTuningDatasetDistributionDict]`

Output only. Dataset distributions for the user output tokens.

pydantic model genai.types.**SupervisedTuningDatasetDistribution**

Bases: `BaseModel`

Dataset distribution for Supervised Tuning.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

Skip to content ly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- billable_sum (int | None)
- buckets (list[genai.types.SupervisedTuningDatasetDistributionDatasetBucket] | None)
- max (float | None)
- mean (float | None)
- median (float | None)
- min (float | None)
- p5 (float | None)
- p95 (float | None)
- sum (int | None)

field billable_sum: Optional[int] = None (alias 'billableSum')

Output only. Sum of a given population of values that are billable.

field buckets: Optional[list[SupervisedTuningDatasetDistributionDatasetBucket]] = None

Output only. Defines the histogram bucket.

field max: Optional[float] = None

Output only. The maximum of the population values.

field mean: Optional[float] = None

Output only. The arithmetic mean of the values in the population.

field median: Optional[float] = None

Output only. The median of the values in the population.

field min: Optional[float] = None

Output only. The minimum of the population values.

field p5: Optional[float] = None

Output only. The 5th percentile of the values in the population.

field p95: Optional[float] = None

Output only. The 95th percentile of the values in the population.

field sum: Optional[int] = None

Output only. Sum of a given population of values.

pydantic model genai.types.**SupervisedTuningDatasetDistributionDatasetBucket**

Bases: BaseModel

Dataset bucket used to create a histogram for the distribution given a population of values.

Create a new model by parsing and validating input data from keyword arguments.

Skip to content

ationError][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `count (float | None)`
- `left (float | None)`
- `right (float | None)`

field count: `Optional[float] = None`

Output only. Number of values in the bucket.

field left: `Optional[float] = None`

Output only. Left bound of the bucket.

field right: `Optional[float] = None`

Output only. Right bound of the bucket.

class genai.types.**SupervisedTuningDatasetDistributionDatasetBucketDict**

Bases: `TypedDict`

Dataset bucket used to create a histogram for the distribution given a population of values.

count: `Optional[float]`

Output only. Number of values in the bucket.

left: `Optional[float]`

Output only. Left bound of the bucket.

right: `Optional[float]`

Output only. Right bound of the bucket.

class genai.types.**SupervisedTuningDatasetDistributionDict**

Skip to content

Bases: `TypedDict`

Dataset distribution for Supervised Tuning.

billable_sum: `Optional[int]`

Output only. Sum of a given population of values that are billable.

buckets: `Optional[list[SupervisedTuningDatasetDistributionDatasetBucketDict]]`

Output only. Defines the histogram bucket.

max: `Optional[float]`

Output only. The maximum of the population values.

mean: `Optional[float]`

Output only. The arithmetic mean of the values in the population.

median: `Optional[float]`

Output only. The median of the values in the population.

min: `Optional[float]`

Output only. The minimum of the population values.

p5: `Optional[float]`

Output only. The 5th percentile of the values in the population.

p95: `Optional[float]`

Output only. The 95th percentile of the values in the population.

sum: `Optional[int]`

Output only. Sum of a given population of values.

pydantic model genai.types.**SupervisedTuningSpec**

Bases: `BaseModel`

Tuning Spec for Supervised Tuning for first party models.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

Skip to content

FIELDS:

- `hyper_parameters (genai.types.SupervisedHyperParameters | None)`
- `training_dataset_uri (str | None)`
- `validation_dataset_uri (str | None)`

field hyper_parameters: `Optional[SupervisedHyperParameters] = None (alias 'hyperParameters')`

Optional. Hyperparameters for SFT.

field training_dataset_uri: `Optional[str] = None (alias 'trainingDatasetUri')`

Required. Cloud Storage path to file containing training dataset for tuning. The dataset must be formatted as a JSONL file.

field validation_dataset_uri: `Optional[str] = None (alias 'validationDatasetUri')`

Optional. Cloud Storage path to file containing validation dataset for tuning. The dataset must be formatted as a JSONL file.

class genai.types.**SupervisedTuningSpecDict**

Bases: `TypedDict`

Tuning Spec for Supervised Tuning for first party models.

hyper_parameters: `Optional[SupervisedHyperParametersDict]`

Optional. Hyperparameters for SFT.

training_dataset_uri: `Optional[str]`

Required. Cloud Storage path to file containing training dataset for tuning. The dataset must be formatted as a JSONL file.

validation_dataset_uri: `Optional[str]`

Optional. Cloud Storage path to file containing validation dataset for tuning. The dataset must be formatted as a JSONL file.

pydantic model genai.types.**TestTableFile**

Bases: `BaseModel`

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

Skip to content

FIELDS:

- `comment (str | None)`
- `parameter_names (list[str] | None)`
- `test_method (str | None)`
- `test_table (list[genai.types.TestTableItem] | None)`

field comment: Optional[str] = None

field parameter_names: Optional[list[str]] = None (alias 'parameterNames')

field test_method: Optional[str] = None (alias 'testMethod')

field test_table: Optional[list[TestTableItem]] = None (alias 'testTable')

class genai.types.**TestTableFileDict**

Bases: `TypedDict`

comment: Optional[str]

parameter_names: Optional[list[str]]

test_method: Optional[str]

test_table: Optional[list[TestTableItemDict]]

pydantic model genai.types.**TestTableItem**

Skip to content

Bases: `BaseModel`

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `exception_if_mldev (str | None)`
- `exception_if_vertex (str | None)`
- `has_union (bool | None)`
- `name (str | None)`
- `override_replay_id (str | None)`
- `parameters (dict[str, Any] | None)`
- `skip_in_api_mode (str | None)`

field exception_if_mldev: `Optional[str] = None (alias 'exceptionIfMldev')`

Expects an exception for MLDev matching the string.

field exception_if_vertex: `Optional[str] = None (alias 'exceptionIfVertex')`

Expects an exception for Vertex matching the string.

field has_union: `Optional[bool] = None (alias 'hasUnion')`

True if the parameters contain an unsupported union type. This test will be skipped for languages that do not support the union type.

field name: `Optional[str] = None`

The name of the test. This is used to derive the replay id.

field override_replay_id: `Optional[str] = None (alias 'overrideReplayId')`

Use if you don't want to use the default replay id which is derived from the test name.

field parameters: `Optional[dict[str, Any]] = None`

The parameters to the test. Use pydantic models.

field skip_in_api_mode: `Optional[str] = None (alias 'skipInApiMode')`

When set to a reason string, this test will be skipped in the API mode. Use this flag for tests that can not be reproduced with the real API. E.g. a test that deletes a resource.

class genai.types.**TestTableItemDict**

Skip to content

Bases: `TypedDict`

exception_if_mldev: `Optional[str]`

Expects an exception for MLDev matching the string.

exception_if_vertex: `Optional[str]`

Expects an exception for Vertex matching the string.

has_union: `Optional[bool]`

True if the parameters contain an unsupported union type. This test will be skipped for languages that do not support the union type.

name: `Optional[str]`

The name of the test. This is used to derive the replay id.

override_replay_id: `Optional[str]`

Use if you don't want to use the default replay id which is derived from the test name.

parameters: `Optional[dict[str, Any]]`

The parameters to the test. Use pydantic models.

skip_in_api_mode: `Optional[str]`

When set to a reason string, this test will be skipped in the API mode. Use this flag for tests that can not be reproduced with the real API. E.g. a test that deletes a resource.

pydantic model genai.types.**ThinkingConfig**

Bases: `BaseModel`

The thinking features configuration.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `include_thoughts (bool | None)`

field include_thoughts: `Optional[bool] = None (alias 'includeThoughts')`

Indicates whether to include thoughts in the response. If true, thoughts are returned only if the model supports thought and thoughts are available.

class genai.types.**ThinkingConfigDict**

Skip to content `Dict`

The thinking features configuration.

include_thoughts: Optional[bool]

Indicates whether to include thoughts in the response. If true, thoughts are returned only if the model supports thought and thoughts are available.

pydantic model genai.types.**TokensInfo**

Bases: BaseModel

Tokens info with a list of tokens and the corresponding list of token ids.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- role (str | None)
- token_ids (list[int] | None)
- tokens (list[bytes] | None)

field role: Optional[str] = None

Optional. Optional fields for the role from the corresponding Content.

field token_ids: Optional[list[int]] = None (alias 'tokenIds')

A list of token ids from the input.

field tokens: Optional[list[bytes]] = None

A list of tokens from the input.

class genai.types.**TokensInfoDict**

Bases: TypedDict

Tokens info with a list of tokens and the corresponding list of token ids.

role: Optional[str]

Optional. Optional fields for the role from the corresponding Content.

token_ids: Optional[list[int]]

A list of token ids from the input.

tokens: Optional[list[bytes]]

A list of tokens from the input.

Skip to content

genai.types.**Tool**

Bases: BaseModel

Tool details of a tool that the model may use to generate a response.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `code_execution (genai.types.ToolCodeExecution | None)`
- `function_declarations (list[genai.types.FunctionDeclaration] | None)`
- `google_search (genai.types.GoogleSearch | None)`
- `google_search_retrieval (genai.types.GoogleSearchRetrieval | None)`
- `retrieval (genai.types.Retrieval | None)`

field code_execution: `Optional[ToolCodeExecution] = None (alias 'codeExecution')`

Optional. CodeExecution tool type. Enables the model to execute code as part of generation. This field is only used by the Gemini Developer API services.

field function_declarations: `Optional[list[FunctionDeclaration]] = None (alias 'functionDeclarations')`

List of function declarations that the tool supports.

field google_search: `Optional[GoogleSearch] = None (alias 'googleSearch')`

Optional. Google Search tool type. Specialized retrieval tool that is powered by Google Search.

field google_search_retrieval: `Optional[GoogleSearchRetrieval] = None (alias 'googleSearchRetrieval')`

Optional. GoogleSearchRetrieval tool type. Specialized retrieval tool that is powered by Google search.

field retrieval: `Optional[Retrieval] = None`

Optional. Retrieval tool type. System will always execute the provided retrieval tool(s) to get external knowledge to answer the prompt. Retrieval results are presented to the model for generation.

pydantic model genai.types.**ToolCodeExecution**

Bases: `BaseModel`

Tool that executes code generated by the model, and automatically returns the result to the model.

cutableCode]and [CodeExecutionResult] which are input and output to this tool.

Skip to content

model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

class genai.types.ToolCodeExecutionDict

Bases: `TypedDict`

Tool that executes code generated by the model, and automatically returns the result to the model.

See also [ExecutableCode]and [CodeExecutionResult] which are input and output to this tool.

pydantic model genai.types.ToolConfig

Bases: `BaseModel`

Tool config.

This config is shared for all tools provided in the request.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `function_calling_config (genai.types.FunctionCallingConfig | None)`

field function_calling_config: Optional[FunctionCallingConfig] = None (alias 'functionCallingConfig')

Optional. Function calling config.

class genai.types.ToolConfigDict

Bases: `TypedDict`

Tool config.

This config is shared for all tools provided in the request.

function_calling_config: Optional[FunctionCallingConfigDict]

Optional. Function calling config.

class genai.types.ToolDict

`Dict`

Skip to content

Tool details of a tool that the model may use to generate a response.

code_execution: Optional[ToolCodeExecutionDict]

Optional. CodeExecution tool type. Enables the model to execute code as part of generation. This field is only used by the Gemini Developer API services.

function_declarations: Optional[list[FunctionDeclarationDict]]

List of function declarations that the tool supports.

google_search: Optional[GoogleSearchDict]

Optional. Google Search tool type. Specialized retrieval tool that is powered by Google Search.

google_search_retrieval: Optional[GoogleSearchRetrievalDict]

Optional. GoogleSearchRetrieval tool type. Specialized retrieval tool that is powered by Google search.

retrieval: Optional[RetrievalDict]

Optional. Retrieval tool type. System will always execute the provided retrieval tool(s) to get external knowledge to answer the prompt. Retrieval results are presented to the model for generation.

pydantic model genai.types.**TunedModel**

Bases: `BaseModel`

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `endpoint (str | None)`
- `model (str | None)`

field endpoint: Optional[str] = None

Output only. A resource name of an Endpoint. Format: *projects/{project}/locations/{location}/endpoints/{endpoint}*.

field model: Optional[str] = None

Output only. The resource name of the TunedModel. Format: *projects/{project}/locations/{location}/models/{model}*.

Skip to content ...es.**TunedModelDict**

Bases: `TypedDict`

endpoint: `Optional[str]`

projects/{project}/locations/{location}/endpoints/{endpoint}.

TYPE:
Output only. A resource name of an Endpoint. Format

model: `Optional[str]`

projects/{project}/locations/{location}/models/{model}.

TYPE:
Output only. The resource name of the TunedModel. Format

pydantic model genai.types.**TunedModelInfo**

Bases: `BaseModel`

A tuned machine learning model.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `base_model (str | None)`
- `create_time (datetime.datetime | None)`
- `update_time (datetime.datetime | None)`

field base_model: `Optional[str] = None (alias 'baseModel')`

ID of the base model that you want to tune.

field create_time: `Optional[datetime] = None (alias 'createTime')`

Date and time when the base model was created.

field update_time: `Optional[datetime] = None (alias 'updateTime')`

Date and time when the base model was last updated.

class genai.types.**TunedModelInfoDict**

Skip to content

Bases: `TypedDict`

A tuned machine learning model.

base_model: `Optional[str]`

ID of the base model that you want to tune.

create_time: `Optional[datetime]`

Date and time when the base model was created.

update_time: `Optional[datetime]`

Date and time when the base model was last updated.

pydantic model genai.types.TuningDataStats

Bases: `BaseModel`

The tuning data statistic values for TuningJob.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `distillation_data_stats (genai.types.DistillationDataStats | None)`
- `supervised_tuning_data_stats (genai.types.SupervisedTuningDataStats | None)`

field distillation_data_stats: `Optional[DistillationDataStats] = None (alias 'distillationDataStats')`

Output only. Statistics for distillation.

field supervised_tuning_data_stats: `Optional[SupervisedTuningDataStats] = None (alias 'supervisedTuningDataStats')`

The SFT Tuning data stats.

class genai.types.TuningDataStatsDict

Bases: `TypedDict`

The tuning data statistic values for TuningJob.

distillation_data_stats: `Optional[DistillationDataStatsDict]`

Output only. Statistics for distillation.

supervised_tuning_data_stats: `Optional[SupervisedTuningDataStatsDict]`

Skip to content Tuning data stats.

pydantic model genai.types.TuningDataset

Bases: `BaseModel`

Supervised fine-tuning training dataset.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `examples (list[genai.types.TuningExample] | None)`
- `gcs_uri (str | None)`

field examples: `Optional[list[TuningExample]] = None`

Inline examples with simple input/output text.

field gcs_uri: `Optional[str] = None (alias 'gcsUri')`

GCS URI of the file containing training dataset in JSONL format.

class genai.types.**TuningDatasetDict**

Bases: `TypedDict`

Supervised fine-tuning training dataset.

examples: `Optional[list[TuningExampleDict]]`

Inline examples with simple input/output text.

gcs_uri: `Optional[str]`

GCS URI of the file containing training dataset in JSONL format.

pydantic model genai.types.**TuningExample**

Skip to content

Bases: `BaseModel`

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `output (str | None)`
- `text_input (str | None)`

field output: `Optional[str] = None`

The expected model output.

field text_input: `Optional[str] = None (alias 'textInput')`

Text model input.

class genai.types.**TuningExampleDict**

Bases: `TypedDict`

output: `Optional[str]`

The expected model output.

text_input: `Optional[str]`

Text model input.

pydantic model genai.types.**TuningJob**

Bases: `BaseModel`

A tuning job.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

Skip to content

FIELDS:

- `base_model (str | None)`
- `create_time (datetime.datetime | None)`
- `description (str | None)`
- `distillation_spec (genai.types.DistillationSpec | None)`
- `encryption_spec (genai.types.EncryptionSpec | None)`
- `end_time (datetime.datetime | None)`
- `error (genai.types.GoogleRpcStatus | None)`
- `experiment (str | None)`
- `labels (dict[str, str] | None)`
- `name (str | None)`
- `partner_model_tuning_spec (genai.types.PartnerModelTuningSpec | None)`
- `pipeline_job (str | None)`
- `start_time (datetime.datetime | None)`
- `state (genai.types.JobState | None)`
- `supervised_tuning_spec (genai.types.SupervisedTuningSpec | None)`
- `tuned_model (genai.types.TunedModel | None)`
- `tuned_model_display_name (str | None)`
- `tuning_data_stats (genai.types.TuningDataStats | None)`
- `update_time (datetime.datetime | None)`

field base_model: `Optional[str] = None (alias 'baseModel')`

The base model that is being tuned, e.g., “gemini-1.0-pro-002”. .

field create_time: `Optional[datetime] = None (alias 'createTime')`

Output only. Time when the TuningJob was created.

field description: `Optional[str] = None`

Optional. The description of the TuningJob.

field distillation_spec: `Optional[DistillationSpec] = None (alias 'distillationSpec')`

Tuning Spec for Distillation.

field encryption_spec: `Optional[EncryptionSpec] = None (alias 'encryptionSpec')`

Customer-managed encryption key options for a TuningJob. If this is set, then all resources created by the TuningJob will be encrypted with the provided encryption key.

field end_time: `Optional[datetime] = None (alias 'endTime')`

Output only. Time when the TuningJob entered any of the following JobStates: *JOB_STATE_SUCCEEDED*, *JOB_STATE_FAILED*, *JOB_STATE_CANCELLED*, *JOB_STATE_EXPIRED*.

Skip to content

field error: `Optional[GoogleRpcStatus] = None`

Output only. Only populated when job's state is *JOB_STATE_FAILED* or *JOB_STATE_CANCELLED*.

field experiment: `Optional[str] = None`

Output only. The Experiment associated with this TuningJob.

field labels: `Optional[dict[str, str]] = None`

Optional. The labels with user-defined metadata to organize TuningJob and generated resources such as Model and Endpoint. Label keys and values can be no longer than 64 characters (Unicode codepoints), can only contain lowercase letters, numeric characters, underscores and dashes. International characters are allowed. See https://goo.gl/xmQnxf for more information and examples of labels.

field name: `Optional[str] = None`

Output only. Identifier. Resource name of a TuningJob. Format: *projects/{project}/locations/{location}/tuningJobs/{tuning_job}*

field partner_model_tuning_spec: `Optional[PartnerModelTuningSpec] = None (alias 'partnerModelTuningSpec')`

Tuning Spec for open sourced and third party Partner models.

field pipeline_job: `Optional[str] = None (alias 'pipelineJob')`

Output only. The resource name of the PipelineJob associated with the TuningJob. Format: *projects/{project}/locations/{location}/pipelineJobs/{pipeline_job}*.

field start_time: `Optional[datetime] = None (alias 'startTime')`

Output only. Time when the TuningJob for the first time entered the *JOB_STATE_RUNNING* state.

field state: `Optional[JobState] = None`

Output only. The detailed state of the job.

field supervised_tuning_spec: `Optional[SupervisedTuningSpec] = None (alias 'supervisedTuningSpec')`

Tuning Spec for Supervised Fine Tuning.

field tuned_model: `Optional[TunedModel] = None (alias 'tunedModel')`

Output only. The tuned model resources associated with this TuningJob.

field tuned_model_display_name: `Optional[str] = None (alias 'tunedModelDisplayName')`

Optional. The display name of the TunedModel. The name can be up to 128 characters long and can consist of any UTF-8 characters.

field tuning_data_stats: `Optional[TuningDataStats] = None (alias 'tuningDataStats')`

nly. The tuning data statistics associated with this TuningJob.

Skip to content

field update_time: `Optional[datetime] = None (alias 'updateTime')`

Output only. Time when the TuningJob was most recently updated.

property **has_ended**: bool

Whether the tuning job has ended.

property **has_succeeded**: bool

Whether the tuning job has succeeded.

class genai.types.**TuningJobDict**

Skip to content

Bases: `TypedDict`

A tuning job.

base_model: `Optional[str]`

The base model that is being tuned, e.g., “gemini-1.0-pro-002”. .

create_time: `Optional[datetime]`

Output only. Time when the TuningJob was created.

description: `Optional[str]`

Optional. The description of the TuningJob.

distillation_spec: `Optional[DistillationSpecDict]`

Tuning Spec for Distillation.

encryption_spec: `Optional[EncryptionSpecDict]`

Customer-managed encryption key options for a TuningJob. If this is set, then all resources created by the TuningJob will be encrypted with the provided encryption key.

end_time: `Optional[datetime]`

JOB_STATE_SUCCEEDED, *JOB_STATE_FAILED*, *JOB_STATE_CANCELLED*, *JOB_STATE_EXPIRED*.

TYPE:

Output only. Time when the TuningJob entered any of the following JobStates

error: `Optional[GoogleRpcStatusDict]`

Output only. Only populated when job’s state is *JOB_STATE_FAILED* or *JOB_STATE_CANCELLED*.

experiment: `Optional[str]`

Output only. The Experiment associated with this TuningJob.

labels: `Optional[dict[str, str]]`

//goo.gl/xmQnxf for more information and examples of labels.

TYPE:

Optional. The labels with user-defined metadata to organize TuningJob and generated resources such as Model and Endpoint. Label keys and values can be no longer than 64 characters (Unicode codepoints), can only contain lowercase letters, numeric characters, underscores and dashes. International characters are allowed. See https

name: `Optional[str]`

projects/{project}/locations/{location}/tuningJobs/{tuning_job}

TYPE:

ɔut only. Identifier. Resource name of a TuningJob. Format

Skip to content

partner_model_tuning_spec: `Optional[PartnerModelTuningSpecDict]`

Tuning Spec for open sourced and third party Partner models.

pipeline_job: Optional[str]

projects/{project}/locations/{location}/pipelineJobs/{pipeline_job}.

TYPE:
Output only. The resource name of the PipelineJob associated with the TuningJob. Format

start_time: Optional[datetime]

Output only. Time when the TuningJob for the first time entered the *JOB_STATE_RUNNING* state.

state: Optional[JobState]

Output only. The detailed state of the job.

supervised_tuning_spec: Optional[SupervisedTuningSpecDict]

Tuning Spec for Supervised Fine Tuning.

tuned_model: Optional[TunedModelDict]

Output only. The tuned model resources associated with this TuningJob.

tuned_model_display_name: Optional[str]

Optional. The display name of the TunedModel. The name can be up to 128 characters long and can consist of any UTF-8 characters.

tuning_data_stats: Optional[TuningDataStatsDict]

Output only. The tuning data statistics associated with this TuningJob.

update_time: Optional[datetime]

Output only. Time when the TuningJob was most recently updated.

pydantic model genai.types.**TuningValidationDataset**

Bases: `BaseModel`

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `gcs_uri (str | None)`

field gcs_uri: Optional[str] = None (alias 'gcsUri')

GCS URI of the file containing validation dataset in JSONL format.

Skip to content

class genai.types.**TuningValidationDatasetDict**

Bases: `TypedDict`

gcs_uri: `Optional[str]`

GCS URI of the file containing validation dataset in JSONL format.

class genai.types.Type(*values)

Bases: `CaseInSensitiveEnum`

Optional. The type of the data.

ARRAY = 'ARRAY'

BOOLEAN = 'BOOLEAN'

INTEGER = 'INTEGER'

NUMBER = 'NUMBER'

OBJECT = 'OBJECT'

STRING = 'STRING'

TYPE_UNSPECIFIED = 'TYPE_UNSPECIFIED'

pydantic model genai.types.UpdateCachedContentConfig

Bases: `BaseModel`

Optional parameters for caches.update method.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `expire_time (datetime.datetime | None)`
- `http_options (genai.types.HttpOptions | None)`
- `ttl (str | None)`

field expire_time: `Optional[datetime]` = None (alias 'expireTime')

Timestamp of when this resource is considered expired.

field http_options: `Optional[HttpOptions]` = None (alias 'httpOptions')

Used to override HTTP request options.

field ttl: `Optional[str]` = None

The TTL for this resource. The expiration time is computed: now + TTL.

Skip to content

...es.UpdateCachedContentConfigDict

Bases: `TypedDict`

Optional parameters for caches.update method.

expire_time: Optional[datetime]

Timestamp of when this resource is considered expired.

http_options: Optional[HttpOptionsDict]

Used to override HTTP request options.

ttl: Optional[str]

now + TTL.

TYPE:

The TTL for this resource. The expiration time is computed

pydantic model genai.types.**UpdateModelConfig**

Bases: BaseModel

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- description (str | None)
- display_name (str | None)
- http_options (genai.types.HttpOptions | None)

field description: Optional[str] = None

field display_name: Optional[str] = None (alias 'displayName')

field http_options: Optional[HttpOptions] = None (alias 'httpOptions')

Used to override HTTP request options.

class genai.types.**UpdateModelConfigDict**

Bases: TypedDict

description: Optional[str]

display_name: Optional[str]

http_options: Optional[HttpOptionsDict]

Used to override HTTP request options.

pydantic model genai.types.**UploadFileConfig**

Skip to content odel

Used to override the default configuration.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `display_name (str | None)`
- `http_options (genai.types.HttpOptions | None)`
- `mime_type (str | None)`
- `name (str | None)`

field display_name: `Optional[str] = None (alias 'displayName')`

Optional display name of the file.

field http_options: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

field mime_type: `Optional[str] = None (alias 'mimeType')`

mime_type: The MIME type of the file. If not provided, it will be inferred from the file extension.

field name: `Optional[str] = None`

The name of the file in the destination (e.g., 'files/sample-image'. If not provided one will be generated.

class genai.types.**UploadFileConfigDict**

Bases: `TypedDict`

Used to override the default configuration.

display_name: `Optional[str]`

Optional display name of the file.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

mime_type: `Optional[str]`

The MIME type of the file. If not provided, it will be inferred from the file extension.

TYPE:

mime_type

Skip to content

`nal[str]`

e of the file in the destination (e.g., 'files/sample-image'. If not provided one will be generated.

pydantic model genai.types.**UpscaleImageConfig**

Bases: `BaseModel`

Configuration for upscaling an image.

For more information on this configuration, refer to the Imagen API reference documentation.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `http_options (genai.types.HttpOptions | None)`
- `include_rai_reason (bool | None)`
- `output_compression_quality (int | None)`
- `output_mime_type (str | None)`

field **http_options**: `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

field **include_rai_reason**: `Optional[bool] = None (alias 'includeRaiReason')`

Whether to include a reason for filtered-out images in the response.

field **output_compression_quality**: `Optional[int] = None (alias 'outputCompressionQuality')`

The level of compression if the `output_mime_type` is `image/jpeg`.

field **output_mime_type**: `Optional[str] = None (alias 'outputMimeType')`

The image format that the output should be saved as.

class genai.types.**UpscaleImageConfigDict**

Skip to content

Bases: `TypedDict`

Configuration for upscaling an image.

For more information on this configuration, refer to the Imagen API reference documentation.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

include_rai_reason: `Optional[bool]`

Whether to include a reason for filtered-out images in the response.

output_compression_quality: `Optional[int]`

The level of compression if the `output_mime_type` is `image/jpeg`.

output_mime_type: `Optional[str]`

The image format that the output should be saved as.

pydantic model genai.types.**UpscaleImageParameters**

Bases: `BaseModel`

User-facing config UpscaleImageParameters.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `config (genai.types.UpscaleImageConfig | None)`
- `image (genai.types.Image | None)`
- `model (str | None)`
- `upscale_factor (str | None)`

field config: `Optional[UpscaleImageConfig] = None`

Configuration for upscaling.

field image: `Optional[Image] = None`

The input image to upscale.

field model: `Optional[str] = None`

The model to use.

ıle_factor: `Optional[str] = None (alias 'upscaleFactor')`

Skip to content

ɔr to upscale the image (x2 or x4).

class genai.types.**UpscaleImageParametersDict**

Bases: `TypedDict`

User-facing config UpscaleImageParameters.

config: `Optional[UpscaleImageConfigDict]`

Configuration for upscaling.

image: `Optional[ImageDict]`

The input image to upscale.

model: `Optional[str]`

The model to use.

upscale_factor: `Optional[str]`

The factor to upscale the image (x2 or x4).

pydantic model genai.types.**UpscaleImageResponse**

Bases: `BaseModel`

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `generated_images (list[genai.types.GeneratedImage] | None)`

field generated_images: `Optional[list[GeneratedImage]] = None (alias 'generatedImages')`

Generated images.

class genai.types.**UpscaleImageResponseDict**

Bases: `TypedDict`

generated_images: `Optional[list[GeneratedImageDict]]`

Generated images.

pydantic model genai.types.**UserContent**

Bases: `Content`

UserContent facilitates the creation of a Content object with a user role.

Skip to content

Example usages:

- Create a user Content object with a string: user_content = UserContent("Why is the sky blue?")
- Create a user Content object with a file data Part object: user_content = UserContent(Part.from_uri(file_uril="gs://bucket/file.txt", mime_type="text/plain"))
- Create a user Content object with byte data Part object: user_content = UserContent(Part.from_bytes(data=b"Hello, World!", mime_type="text/plain"))

  You can create a user Content object using other classmethods in the Part class as well. You can also create a user Content using a list of Part objects or strings.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `parts (list[genai.types.Part])`
- `role (Literal['user'])`

field parts: `list[Part]` **[Required]**

field role: `Literal['user']` **= 'user'**

pydantic model genai.types.**VertexAISearch**

Bases: `BaseModel`

Retrieve from Vertex AI Search datastore for grounding.

See https://cloud.google.com/products/agent-builder

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `datastore (str | None)`

field datastore: `Optional[str]` **= None**

Required. Fully-qualified Vertex AI Search data store resource ID. Format: *projects/{project}/locations/{location}/collections/{collection}/dataStores/{dataStore}*

Skip to content

ypes.**VertexAISearchDict**

Bases: `TypedDict`

Retrieve from Vertex AI Search datastore for grounding.

See https://cloud.google.com/products/agent-builder

datastore: `Optional[str]`

projects/{project}/locations/{location}/collections/{collection}/dataStores/{dataStore}

TYPE:

Required. Fully-qualified Vertex AI Search data store resource ID. Format

pydantic model genai.types.**VertexRagStore**

Bases: `BaseModel`

Retrieve from Vertex RAG Store for grounding.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `rag_corpora (list[str] | None)`
- `rag_resources (list[genai.types.VertexRagStoreRagResource] | None)`
- `similarity_top_k (int | None)`
- `vector_distance_threshold (float | None)`

field rag_corpora: `Optional[list[str]] = None (alias 'ragCorpora')`

Optional. Deprecated. Please use rag_resources instead.

field rag_resources: `Optional[list[VertexRagStoreRagResource]] = None (alias 'ragResources')`

Optional. The representation of the rag source. It can be used to specify corpus only or ragfiles. Currently only support one corpus or multiple files from one corpus. In the future we may open up multiple corpora support.

field similarity_top_k: `Optional[int] = None (alias 'similarityTopK')`

Optional. Number of top k results to return from the selected corpora.

field vector_distance_threshold: `Optional[float] = None (alias 'vectorDistanceThreshold')`

Optional. Only return results with vector distance smaller than the threshold.

class genai.types.**VertexRagStoreDict**

Skip to content `Dict`

Retrieve from Vertex RAG Store for grounding.

rag_corpora: `Optional[list[str]]`

Optional. Deprecated. Please use rag_resources instead.

rag_resources: `Optional[list[VertexRagStoreRagResourceDict]]`

Optional. The representation of the rag source. It can be used to specify corpus only or ragfiles. Currently only support one corpus or multiple files from one corpus. In the future we may open up multiple corpora support.

similarity_top_k: `Optional[int]`

Optional. Number of top k results to return from the selected corpora.

vector_distance_threshold: `Optional[float]`

Optional. Only return results with vector distance smaller than the threshold.

pydantic model genai.types.**VertexRagStoreRagResource**

Bases: `BaseModel`

The definition of the Rag resource.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `rag_corpus (str | None)`
- `rag_file_ids (list[str] | None)`

field rag_corpus: `Optional[str] = None (alias 'ragCorpus')`

Optional. RagCorpora resource name. Format: *projects/{project}/locations/{location}/ragCorpora/{rag_corpus}*

field rag_file_ids: `Optional[list[str]] = None (alias 'ragFileIds')`

Optional. rag_file_id. The files should be in the same rag_corpus set in rag_corpus field.

class genai.types.**VertexRagStoreRagResourceDict**

Skip to content

Bases: `TypedDict`

The definition of the Rag resource.

rag_corpus: `Optional[str]`

projects/{project}/locations/{location}/ragCorpora/{rag_corpus}

TYPE:
Optional. RagCorpora resource name. Format

rag_file_ids: `Optional[list[str]]`

Optional. rag_file_id. The files should be in the same rag_corpus set in rag_corpus field.

pydantic model genai.types.**Video**

Skip to content

Bases: `BaseModel`

A generated video.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `mime_type (str | None)`
- `uri (str | None)`
- `video_bytes (bytes | None)`

field mime_type: `Optional[str] = None (alias 'mimeType')`

Video encoding, for example “video/mp4”.

field uri: `Optional[str] = None`

Path to another storage.

field video_bytes: `Optional[bytes] = None (alias 'videoBytes')`

Video bytes.

save(path)

Saves the video to a file.

RETURN TYPE:

None

PARAMETERS:

path – Local path where to save the video.

show()

Shows the video.

If the video has no mime_type, it is assumed to be video/mp4.

This method only works in a notebook environment.

class genai.types.**VideoDict**

Skip to content

Bases: `TypedDict`

A generated video.

mime_type: `Optional[str]`

Video encoding, for example “video/mp4”.

uri: `Optional[str]`

Path to another storage.

video_bytes: `Optional[bytes]`

Video bytes.

pydantic model genai.types.VideoMetadata

Bases: `BaseModel`

Metadata describes the input video content.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `end_offset (str | None)`
- `start_offset (str | None)`

field end_offset: `Optional[str] = None (alias 'endOffset')`

Optional. The end offset of the video.

field start_offset: `Optional[str] = None (alias 'startOffset')`

Optional. The start offset of the video.

class genai.types.VideoMetadataDict

Bases: `TypedDict`

Metadata describes the input video content.

end_offset: `Optional[str]`

Optional. The end offset of the video.

start_offset: `Optional[str]`

Optional. The start offset of the video.

Skip to content

genai.types.VoiceConfig

Bases: `BaseModel`

The configuration for the voice to use.

Create a new model by parsing and validating input data from keyword arguments.

Raises [*ValidationError*][pydantic_core.ValidationError] if the input data cannot be validated to form a valid model.

self is explicitly positional-only to allow *self* as a field name.

▶ Show JSON schema

FIELDS:

- `prebuilt_voice_config (genai.types.PrebuiltVoiceConfig | None)`

field **prebuilt_voice_config**: `Optional`[**PrebuiltVoiceConfig**] = **None** (**alias** '**prebuiltVoiceConfig**')

The configuration for the speaker to use.

class genai.types.**VoiceConfigDict**

Bases: `TypedDict`

The configuration for the voice to use.

prebuilt_voice_config: `Optional`[**PrebuiltVoiceConfigDict**]

The configuration for the speaker to use.