

Submodules

genai.client module

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

Bases: `object`

Client for making asynchronous (non-blocking) requests.

```
property auth_tokens: AsyncTokens
```

```
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. Find your Google Cloud project ID.

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.

Skip to content **s**

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:

```
from google import genai

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

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. Typically used when running test code.

Skip to content

`http_options` (`Union[HttpOptions, HttpOptionsDict]`) – Http options to use for the client.

property `aio`: AsyncClient

```
property auth_tokens: Tokens  
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.

pydantic model 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/456” or “456”

when project and location are initialized in the Vertex AI client. Or “batches/abc” using the Gemini Developer AI 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** – The source of the batch job. Currently Vertex AI supports GCS URI(-s) or BigQuery URI. Example: “gs://path/to/input/data” or “bq://projectId.bqDatasetId.bqTableId”. Gemini Develop API supports List of inlined_request, or file name. Example: “files/file_name”.
- **config** (`CreateBatchJobConfig`) – Optional configuration for the batch job.

RETURNS:

A `BatchJob` object that contains details about the batch job.

Skip to content

```
batch_job = await client.aio.batches.create(  
    model="gemini-2.0-flash-001",  
    src="gs://path/to/input/data",  
)
```

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

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 Vertex AI client. Or “batches/abc” using the Gemini Developer AI client.

[Skip to content](#)

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

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/456” or “456”

when project and location are initialized in the Vertex AI client. Or “batches/abc” using the Gemini Developer AI 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** – The source of the batch job. Currently Vertex AI supports GCS URI(-s) or BigQuery URI. Example: “gs://path/to/input/data” or “bq://projectId.bqDatasetId.bqTableId”. Gemini Developer API supports List of inlined_request, or file name. Example: “files/file_name”.
- **config** (`CreateBatchJobConfig`) – Optional configuration for the batch job.

RETURNS:

A `BatchJob` object that contains details about the batch job.

Skip to content

```
batch_job = client.batches.create(  
    model="gemini-2.0-flash-001",  
    src="gs://path/to/input/data",  
)  
print(batch_job.state)
```

delete(*, name, config=None)

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)

[Skip to content](#)

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 Vertex AI client. Or “batches/abc” using the Gemini Developer AI 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)

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`

Skip to content `..., config=None)`

TYPE:

`AsyncPager [CachedContent]`

async update(*, name, config=None)

Updates 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)`

----- TYPE:

Skip to content `er [CachedContent]`

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

Updates cached content configurations.

```
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-2.0-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:

`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:

```
    its.AsyncChats(modules)
```

Skip to content

at

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-2.0-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.

RETURN TYPE:

`Iterator[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:

```
chat = client.chats.create(model='gemini-2.0-flash')
for chunk in chat.send_message_stream('tell me a story'):
    print(chunk.text)
```

Skip to content

`class genai.chats.Chats(modules)`

Bases: `object`

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 a remotely stored file.

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 does 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`

Skip to content

```

for file in 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)`

[Skip to content](#)

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 *display_name*, *mime_type*, and *name*.

```
class genai.files.Files(api_client_)
```

[Skip to content](#)

Bases: `BaseModule`

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

Deletes a remotely stored file.

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:

Skip to content

```
for file in 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)

[Skip to content](#)

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

RETURN TYPE:

`File`

PARAMETERS:

- **file** – A path to the file or an *I*OB_{ase} object to be uploaded. If it's an I_OBase 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 *display_name*, *mime_type*, and *name*.

[Skip to content](#)

genai.live module

[Preview] Live API client.

```
class genai.live.AsyncLive(api_client)
```

Bases: `BaseModule`

[Preview] AsyncLive.

```
connect(*, model, config=None)
```

[Preview] Connect to the live server.

Note: the live API is currently in preview.

Usage:

```
client = genai.Client(api_key=API_KEY)
config = {}
async with client.aio.live.connect(model='...', config=config) as session:
    await session.send_client_content(
        turns=types.Content(
            role='user',
            parts=[types.Part(text='hello!')])
    ),
    turn_complete=True
)
async for message in session.receive():
    print(message)
```

RETURN TYPE:

`AsyncIterator[AsyncSession]`

PARAMETERS:

- **model** – The model to use for the live session.
- **config** – The configuration for the live session.
- ****kwargs** – additional keyword arguments.

YIELDS:

An `AsyncSession` object.

```
property music: AsyncLiveMusic
```

Skip to content `'e.AsyncSession(api_client, websocket)`

Bases: `object`

[Preview] AsyncSession.

async close()

RETURN TYPE:

None

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.

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)

[Deprecated] Send input to the model.

> **Warning:** This method is deprecated and will be removed in a future version (not before Q3 2025). Please use one of the more specific methods: `send_client_content`, `send_realtime_input`, or `send_tool_response` instead.

The method will send the input request to the server.

RETURN TYPE:

None

PARAMETERS:

- **input** – The input request to the model.

Skip to content **.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 send_client_content(*, turns=None, turn_complete=True)

```

Send non-realtime, turn based content to the model.

There are two ways to send messages to the live API: *send_client_content* and *send_realtime_input*.

send_client_content messages are added to the model context **in order**. Having a conversation using *send_client_content* messages is roughly equivalent to using the *Chat.send_message_stream* method, except that the state of the *chat* history is stored on the API server.

Because of *send_client_content*'s order guarantee, the model cannot respond as quickly to *send_client_content* messages as to *send_realtime_input* messages. This makes the biggest difference when sending objects that have significant preprocessing time (typically images).

The *send_client_content* message sends a list of *Content* objects, which has more options than the *media:Blob* sent by *send_realtime_input*.

The main use-cases for *send_client_content* over *send_realtime_input* are:

- Prefilling a conversation context (including sending anything that can't be represented as a realtime message), before starting a realtime conversation.
- Conducting a non-realtime conversation, similar to *client.chat*, using the live api.

Caution: Interleaving *send_client_content* and *send_realtime_input*

in the same conversation is not recommended and can lead to unexpected results.

PARAMETERS:

- **turns** – A *Content* object or list of *Content* objects (or equivalent dicts).
- **turn_complete** – if true (the default) the model will reply immediately. If false, the model will wait for you to send additional *client_content*, and will not return until you send *turn_complete=True*.

[Skip to content](#)

```
Example: `` import google.genai from google.genai import types import os
```

```
if os.environ.get('GOOGLE_GENAI_USE_VERTEXAI'):
```

```
    MODEL_NAME = 'gemini-2.0-flash-live-preview-04-09'
```

```
else:
```

```
    MODEL_NAME = 'gemini-live-2.5-flash-preview';
```

```
client = genai.Client() async with client.aio.live.connect(
```

```
    model=MODEL_NAME, config={"response_modalities": ["TEXT"]}
```

```
) as session:
```

```
    await session.send_client_content(
```

```
        turns=types.Content(
```

```
            role='user', parts=[types.Part(text="Hello world!")]))
```

```
    async for msg in session.receive():
```

```
        if msg.text:
```

```
            print(msg.text)
```

RETURN TYPE:

None

```
async send_realtime_input(*, media=None, audio=None, audio_stream_end=None,
                           video=None, text=None, activity_start=None, activity_end=None)
```

Send realtime input to the model, only send one argument per call.

Use `send_realtime_input` for realtime audio chunks and video frames(images).

With `send_realtime_input` the api will respond to audio automatically based on voice activity detection (VAD).

`send_realtime_input` is optimized for responsiveness at the expense of deterministic ordering. Audio and video tokens are added to the context when they become available.

RETURN TYPE:

None

PARAMETERS:

Skip to content `ia` – A *Blob-like* object, the realtime media to send.

Example: `` from pathlib import Path

```
from google import genai from google.genai import types  
import PIL.Image  
import os  
  
if os.environ.get('GOOGLE_GENAI_USE_VERTEXAI'):  
    MODEL_NAME = 'gemini-2.0-flash-live-preview-04-09'  
  
else:  
    MODEL_NAME = 'gemini-live-2.5-flash-preview';  
  
client = genai.Client()  
  
async with client.aio.live.connect(  
    model=MODEL_NAME, config={"response_modalities": ["TEXT"]},  
) as session:  
    await session.send_realtime_input(  
        media=PIL.Image.open('image.jpg'))  
        audio_bytes = Path('audio.pcm').read_bytes() await session.send_realtime_input(  
  
            media=types.Blob(data=audio_bytes, mime_type='audio/pcm;rate=16000'))
```

async for msg in session.receive():

```
    if msg.text is not None:  
        print(f'{msg.text}')  
  
    ...
```

async **send_tool_response**(*, function_responses)

Send a tool response to the session.

Use *send_tool_response* to reply to *LiveServerToolCall* messages from the server.

To set the available tools, use the *config.tools* argument when you connect to the session (*client.live.connect*).

RETURN TYPE:

None

PARAMETERS:

Skip to content **function_responses** – A *FunctionResponse*-like object or list of *FunctionResponse*-like objects.

Example: `from google import genai from google.genai import types`

```
import os

if os.environ.get('GOOGLE_GENAI_USE_VERTEXAI'):
    MODEL_NAME = 'gemini-2.0-flash-live-preview-04-09'

else:
    MODEL_NAME = 'gemini-live-2.5-flash-preview'

client = genai.Client()

tools = [{"function_declarations": [{"name": "turn_on_the_lights"}]}] config = {

    "tools": tools, "response_modalities": ['TEXT']
```

[Skip to content](#)

```
    }

async with client.aio.live.connect(
    model='models/gemini-live-2.5-flash-preview', config=config

) as session:
    prompt = "Turn on the lights please" await session.send_client_content(
        turns={"parts": [{"text": prompt}]}

)

async for chunk in session.receive():
    if chunk.server_content:
        if chunk.text is not None:
            print(chunk.text)

    elif chunk.tool_call:
        print(chunk.tool_call) print('_'*80)
        function_response=types.FunctionResponse(
            name='turn_on_the_lights', response={'result': 'ok'},
            id=chunk.tool_call.function_calls[0].id,
        )

        print(function_response) await session.send_tool_response(
            function_responses=function_response
        )

    print('_'*80)

async start_stream(*, stream, mime_type)
```

[DRAFTED] Start a live session from a data stream.

Skip to content

> **Warning:** This method is deprecated and will be removed in a future version (not before Q2 2025). Please use one of the more specific methods: `send_client_content`, `send_realtime_input`, or `send_tool_response` instead.

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.

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?',
```

Skip to content `response`)

```
    # local_tokens=5 cached_content_token_count=None
```

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

RETURN TYPE:

DeleteModelResponse

```
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,
    )
)
```

Skip to content `se.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.
- **contents** (*list[Content]*) – The contents to embed.
- **config** (*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.

Built-in MCP support is an experimental feature.

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 = 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.

For the `model` parameter, supported formats for Vertex AI API include: - 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'

Skip to content `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.

Built-in MCP support is an experimental feature.

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 await 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.
```

RETURN TYPE:

`abc.Iterator[GenerateContentResponse]`

Skip to content `__aite_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.
```

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

Generates videos based on an input (text, image, or video) and configuration.

The following use cases are supported: 1. Text to video generation. 2a. Image to video generation (additional text prompt is optional). 2b. Image to video generation with frame interpolation (specify last_frame in config). 3. Video extension (additional text prompt is optional)

RETURN TYPE:

GenerateVideosOperation

PARAMETERS:

- **model** – The model to use.
- **prompt** – The text prompt for generating the videos. Optional for image to video use cases.
- **image** – The input image for generating the videos. Optional if prompt is provided.
- **video** – The input video for video extension use cases. Optional if prompt or image is provided.

Skip to content

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:**

```

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' ...
```

Skip to content

**e(\*, 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:

Skip to content

```
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
```

**delete**(\*, model, config=None)

RETURN TYPE:

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:

[Skip to content](#)

```
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:

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

Usage:

[Skip to content](#)

```
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: - 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.

Built-in MCP support is an experimental feature.

[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(text='What is shown in this image?'),
 types.Part.from_uri(file_uri='gs://generativeai-downloads/images/scones.jpg',
 mime_type='image/jpeg')
]
)
print(response.text)
The image shows a flat lay arrangement of freshly baked blueberry
scones.
```

#### RETURN TYPE:

GenerateContentResponse

**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: - The Gemini model ID, for example: 'gemini-2.0-flash' - The full resource name starts with 'projects/', for example:

[Skip to content](#)

'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.

Built-in MCP support is an experimental feature.

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'
)

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.

```

**RETURN TYPE:**

`Iterator[GenerateContentResponse]`

**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](#)

```
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, video=None, config=None)

Generates videos based on an input (text, image, or video) and configuration.

The following use cases are supported: 1. Text to video generation. 2a. Image to video generation (additional text prompt is optional). 2b. Image to video generation with frame interpolation (specify last\_frame in config). 3. Video extension (additional text prompt is optional)

### RETURN TYPE:

GenerateVideosOperation

### PARAMETERS:

- **model** – The model to use.
- **prompt** – The text prompt for generating the videos. Optional for image to video use cases.
- **image** – The input image for generating the videos. Optional if prompt is provided.
- **video** – The input video for video extension use cases. Optional if prompt or image is provided.
- **config** – Configuration for generation.

### Usage:

[Skip to content](#)

```

    ```` 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.

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)**Skip to content TYPE:**`...uel`

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.tokens module

[Experimental] Auth Tokens API client.

```
class genai.tokens.AsyncTokens(api_client_)
```

Bases: `BaseModule`

[Experimental] Async Auth Tokens API client.

This class provides asynchronous methods for creating auth tokens.

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

Creates an auth token asynchronously. Support in v1alpha only.

RETURN TYPE:

`AuthToken`

PARAMETERS:

config (*CreateAuthTokenConfig*) – Optional configuration for the request.

Usage:

```
class genai.tokens.Tokens(api_client_)
```

Bases: `BaseModule`

[Experimental] Auth Tokens API client.

[Skip to content](#)

This class provides methods for creating auth tokens.

create(*, config=None)

[Experimental] Creates an auth token.

RETURN TYPE:

AuthToken

PARAMETERS:

config (*CreateAuthTokenConfig*) – Optional configuration for the request.

The *CreateAuthTokenConfig*'s *live_constrained_parameters* attribute can be used to lock the parameters of the live session so they can't be changed client side. This behavior has two basic modes depending on whether *lock_additional_fields* is set:

If you do not pass *lock_additional_fields* the entire *live_constrained_parameters* is locked and can't be changed by the token's user.

If you set *lock_additional_fields*, then the non-null fields of *live_constrained_parameters* are locked, and any additional fields specified in *lock_additional_fields*.

Usage:

```
# Case 1: If LiveEphemeralParameters is unset, unlock LiveConnectConfig
# when using the token in Live API sessions. Each session connection can
# use a different configuration.

config = types.CreateAuthTokenConfig(
    uses=10,
    expire_time='2025-05-01T00:00:00Z',
)
auth_token = client.tokens.create(config=config)
```

Skip to content

```
# Case 2: If LiveEphemeralParameters is set, lock all fields in
# LiveConnectConfig when using the token in Live API sessions. For
# example, changing `output_audio_transcription` in the Live API
# connection will be ignored by the API.
```

```
auth_token = client.tokens.create(
    config=types.CreateAuthTokenConfig(
        uses=10,
        live_constrained_parameters=types.LiveEphemeralParameters(
            model='gemini-live-2.5-flash-preview',
            config=types.LiveConnectConfig(
                system_instruction='You are an LLM called Gemini.'
            ),
        ),
    )
)
```

```
# Case 3: If LiveEphemeralParameters is set and lockAdditionalFields is
# empty, lock LiveConnectConfig with set fields (e.g.
# system_instruction in this example) when using the token in Live API
# sessions.
```

```
auth_token = client.tokens.create(
    config=types.CreateAuthTokenConfig(
        uses=10,
        live_constrained_parameters=types.LiveEphemeralParameters(
            config=types.LiveConnectConfig(
                system_instruction='You are an LLM called Gemini.'
            ),
        ),
        lock_additional_fields=[],
    )
)
```

[Skip to content](#)

```
# Case 4: If LiveEphemeralParameters is set and lockAdditionalFields is set, lock LiveConnectConfig with set and additional fields (e.g. system_instruction, temperature in this example) when using the token # in Live API sessions.
auth_token = client.tokens.create(
    config=types.CreateAuthTokenConfig(
        uses=10,
        live_constrained_parameters=types.LiveEphemeralParameters(
            model='gemini-live-2.5-flash-preview',
            config=types.LiveConnectConfig(
                system_instruction='You are an LLM called Gemini.'
            ),
        ),
        lock_additional_fields=['temperature'],
    )
)
```

[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

pydantic model genai.types.ActivityEnd

Bases: `BaseModel`

Marks the end of user activity.

This can only be sent if automatic (i.e. server-side) activity detection is disabled.

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.ActivityEndDict

Bases: `TypedDict`

Marks the end of user activity.

This can only be sent if automatic (i.e. server-side) activity detection is disabled.

class genai.types.ActivityHandling(*values)

Bases: `CaseInsensitiveEnum`

The different ways of handling user activity.

ACTIVITY_HANDLING_UNSPECIFIED = `'ACTIVITY_HANDLING_UNSPECIFIED'`

If unspecified, the default behavior is `START_OF_ACTIVITY_INTERRUPTS`.

NO INTERRUPTION = `'NO INTERRUPTION'`

The model's response will not be interrupted.

START_OF_ACTIVITY_INTERRUPTS = `'START_OF_ACTIVITY_INTERRUPTS'`

If true, start of activity will interrupt the model's response (also called "barge in"). The model's current response will be cut-off in the moment of the interruption. This is the default behavior.

pydantic model genai.types.ActivityStart

Skip to content `odel`

Marks the start of user activity.

This can only be sent if automatic (i.e. server-side) activity detection is disabled.

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.ActivityStartDict`

Bases: `TypedDict`

Marks the start of user activity.

This can only be sent if automatic (i.e. server-side) activity detection is disabled.

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

Bases: `CaseInsensitiveEnum`

Optional. Adapter size for tuning.

`ADAPTER_SIZE_EIGHT` = 'ADAPTER_SIZE_EIGHT'

Adapter size 8.

`ADAPTER_SIZE_FOUR` = 'ADAPTER_SIZE_FOUR'

Adapter size 4.

`ADAPTER_SIZE_ONE` = 'ADAPTER_SIZE_ONE'

Adapter size 1.

`ADAPTER_SIZE_SIXTEEN` = 'ADAPTER_SIZE_SIXTEEN'

Adapter size 16.

`ADAPTER_SIZE_THIRTY_TWO` = 'ADAPTER_SIZE_THIRTY_TWO'

Adapter size 32.

`ADAPTER_SIZE_TWO` = 'ADAPTER_SIZE_TWO'

Adapter size 2.

`ADAPTER_SIZE_UNSPECIFIED` = 'ADAPTER_SIZE_UNSPECIFIED'

Skip to content size is unspecified.

`pydantic model genai.types.ApiAuth`

Bases: `BaseModel`

The generic reusable api auth config.

Deprecated. Please use AuthConfig (google/cloud/aiplatform/master/auth.proto) instead.

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_key_config` (`genai.types.ApiAuthApiKeyConfig` | `None`)

field `api_key_config`: `Optional[ApiAuthApiKeyConfig] = None (alias 'apiKeyConfig')`

The API secret.

pydantic model `genai.types.ApiAuthApiKeyConfig`

Bases: `BaseModel`

The API secret.

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_key_secret_version` (`str` | `None`)
- `api_key_string` (`str` | `None`)

field `api_key_secret_version`: `Optional[str] = None (alias 'apiKeySecretVersion')`

Required. The SecretManager secret version resource name storing API key. e.g.
`projects/{project}/secrets/{secret}/versions/{version}`

field `api_key_string`: `Optional[str] = None (alias 'apiKeyString')`

The API key string. Either this or `api_key_secret_version` must be set.

Skip to content [`es.ApiAuthApiKeyConfigDict`](#)

Bases: `ItypedDict`

The API secret.

api_key_secret_version: `Optional[str]`

Required. The SecretManager secret version resource name storing API key. e.g. `projects/{project}/secrets/{secret}/versions/{version}`

api_key_string: `Optional[str]`

The API key string. Either this or `api_key_secret_version` must be set.

class genai.types.ApiAuthDict

Bases: `TypedDict`

The generic reusable api auth config.

Deprecated. Please use `AuthConfig` (`google/cloud/aiplatform/master/auth.proto`) instead.

api_key_config: `Optional[ApiKeyConfigDict]`

The API secret.

pydantic model genai.types.ApiKeyConfig

Bases: `BaseModel`

Config for authentication with API key.

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_key_string (str | None)`

field api_key_string: Optional[str] = None (alias 'apiKeyString')

The API key to be used in the request directly.

class genai.types.ApiKeyConfigDict

Bases: `TypedDict`

Skip to content

- `api_key_string: Optional[str]`

The API key to be used in the request directly.

class genai.types.ApiSpec(*values)

Bases: CaseInsensitiveEnum

The API spec that the external API implements.

API_SPEC_UNSPECIFIED = 'API_SPEC_UNSPECIFIED'

Unspecified API spec. This value should not be used.

ELASTIC_SEARCH = 'ELASTIC_SEARCH'

Elastic search API spec.

SIMPLE_SEARCH = 'SIMPLE_SEARCH'

Simple search API spec.

pydantic model genai.types.AudioChunk

Bases: BaseModel

Representation of an audio 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:

- `data` (bytes | None)
- `mime_type` (str | None)
- `source_metadata` (genai.types.LiveMusicSourceMetadata | None)

field data: Optional[bytes] = None

Raw bytes of audio data.

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

MIME type of the audio chunk.

field source_metadata: Optional[LiveMusicSourceMetadata] = None (alias 'sourceMetadata')

and config used for generating this audio chunk.

Skip to content

class genai.types.AudioChunkDict

Bases: TypedDict

Representation of an audio chunk.

data: Optional [bytes]

Raw bytes of audio data.

mime_type: Optional [str]

MIME type of the audio chunk.

source_metadata: Optional [LiveMusicSourceMetadataDict]

Prompts and config used for generating this audio chunk.

pydantic model genai.types.AudioTranscriptionConfig

Bases: BaseModel

The audio transcription configuration in Setup.

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.AudioTranscriptionConfigDict

Bases: TypedDict

The audio transcription configuration in Setup.

pydantic model genai.types.AuthConfig

Bases: BaseModel

Auth configuration to run the extension.

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:

- `api_key_config` (`genai.types.ApiKeyConfig | None`)
- `auth_type` (`genai.types.AuthType | None`)
- `google_service_account_config` (`genai.types.AuthConfigGoogleServiceAccountConfig | None`)
- `http_basic_auth_config` (`genai.types.AuthConfigHttpBasicAuthConfig | None`)
- `oauth_config` (`genai.types.AuthConfigOauthConfig | None`)
- `oidc_config` (`genai.types.AuthConfigOidcConfig | None`)

field `api_key_config`: `Optional[ApiKeyConfig] = None (alias 'apiKeyConfig')`

Config for API key auth.

field `auth_type`: `Optional[AuthType] = None (alias 'authType')`

Type of auth scheme.

field `google_service_account_config`: `Optional[AuthConfigGoogleServiceAccountConfig] = None (alias 'googleServiceAccountConfig')`

Config for Google Service Account auth.

field `http_basic_auth_config`: `Optional[AuthConfigHttpBasicAuthConfig] = None (alias 'httpBasicAuthConfig')`

Config for HTTP Basic auth.

field `oauth_config`: `Optional[AuthConfigOauthConfig] = None (alias 'oauthConfig')`

Config for user oauth.

field `oidc_config`: `Optional[AuthConfigOidcConfig] = None (alias 'oidcConfig')`

Config for user OIDC auth.

class `genai.types.AuthConfigDict`

Skip to content

Bases: `TypedDict`

Auth configuration to run the extension.

`api_key_config: Optional[ApiKeyConfigDict]`

Config for API key auth.

`auth_type: Optional[AuthType]`

Type of auth scheme.

`google_service_account_config: Optional[AuthConfigGoogleServiceAccountConfigDict]`

Config for Google Service Account auth.

`http_basic_auth_config: Optional[AuthConfigHttpBasicAuthConfigDict]`

Config for HTTP Basic auth.

`oauth_config: Optional[AuthConfigOAuthConfigDict]`

Config for user oauth.

`oidc_config: Optional[AuthConfigOIDCConfigDict]`

Config for user OIDC auth.

pydantic model `genai.types.AuthConfigGoogleServiceAccountConfig`

Bases: `BaseModel`

Config for Google Service Account Authentication.

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:

- `service_account (str | None)`

`field service_account: Optional[str] = None (alias 'serviceAccount')`

Optional. The service account that the extension execution service runs as. - If the service account is specified, the `iam.serviceAccounts.getAccessToken` permission should be

Skip to content o Vertex AI Extension Service Agent (<https://cloud.google.com/vertex-general/access-control#service-agents>) on the specified service account. - If not specified, the Vertex AI Extension Service Agent will be used to execute the Extension.

class genai.types.AuthConfigGoogleServiceAccountConfigDictBases: `TypedDict`

Config for Google Service Account Authentication.

`service_account: Optional[str]`

//cloud.google.com/vertex-ai/docs/general/access-control#service-agents) on the specified service account. - If not specified, the Vertex AI Extension Service Agent will be used to execute the Extension.

TYPE:

Optional. The service account that the extension execution service runs as. - If the service account is specified, the *iam.serviceAccounts.getAccessToken* permission should be granted to Vertex AI Extension Service Agent (<https://cloud.google.com/vertex-ai/docs/general/access-control#service-agents>)

pydantic model genai.types.AuthConfigHttpBasicAuthConfigBases: `BaseModel`

Config for HTTP Basic Authentication.

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:

- `credential_secret (str | None)`

`field credential_secret: Optional[str] = None (alias 'credentialSecret')`

Required. The name of the SecretManager secret version resource storing the base64 encoded credentials. Format: *projects/{project}/secrets/{secret}/versions/{version}* - If specified, the *secretmanager.versions.access* permission should be granted to Vertex AI Extension Service Agent (<https://cloud.google.com/vertex-ai/docs/general/access-control#service-agents>) on the specified resource.

class genai.types.AuthConfigHttpBasicAuthConfigDictBases: `TypedDict`

Skip to content

Config for HTTP Basic Authentication.

credential_secret: Optional [str]

//cloud.google.com/vertex-ai/docs/general/access-control#service-agents) on the specified resource.

TYPE:

Required. The name of the SecretManager secret version resource storing the base64 encoded credentials. Format

TYPE:

`projects/{project}/secrets/{secret}/versions/{version}` - If specified, the `secretmanager.versions.access` permission should be granted to Vertex AI Extension Service Agent (<https://cloud.google.com/vertex-ai/docs/general/access-control#service-agents>)

pydantic model `genai.types.AuthConfigOauthConfig`

Bases: `BaseModel`

Config for user oauth.

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:

- `access_token` (str | None)
- `service_account` (str | None)

field `access_token`: Optional [str] = None (alias 'accessToken')

Access token for extension endpoint. Only used to propagate token from [[`ExecuteExtensionRequest.runtime_auth_config`]] at request time.

field `service_account`: Optional [str] = None (alias 'serviceAccount')

The service account used to generate access tokens for executing the Extension. - If the service account is specified, the `iam.serviceAccounts.getAccessToken` permission should be granted to Vertex AI Extension Service Agent (<https://cloud.google.com/vertex-ai/docs/general/access-control#service-agents>) on the provided service account.

Skip to content

`, pes.AuthConfigOauthConfigDict`

Bases: `TypedDict`

Config for user oauth.

access_token: `Optional[str]`

Access token for extension endpoint. Only used to propagate token from `[[ExecuteExtensionRequest.runtime_auth_config]]` at request time.

service_account: `Optional[str]`

`//cloud.google.com/vertex-ai/docs/general/access-control#service-agents` on the provided service account.

TYPE:

The service account used to generate access tokens for executing the Extension. - If the service account is specified, the `iam.serviceAccounts.getAccessToken` permission should be granted to Vertex AI Extension Service Agent (<https://cloud.google.com/vertex-ai/docs/general/access-control#service-agents>)

pydantic model `genai.types.AuthConfigOidcConfig`

Bases: `BaseModel`

Config for user OIDC auth.

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:

- `id_token (str | None)`
- `service_account (str | None)`

field id_token: `Optional[str] = None (alias 'idToken')`

OpenID Connect formatted ID token for extension endpoint. Only used to propagate token from `[[ExecuteExtensionRequest.runtime_auth_config]]` at request time.

field service_account: `Optional[str] = None (alias 'serviceAccount')`

The service account used to generate an OpenID Connect (OIDC)-compatible JWT token signed by the Google OIDC Provider (`accounts.google.com`) for extension endpoint (<https://cloud.google.com/iam/docs/create-short-lived-credentials-direct#sa-credentials>).

The audience for the token will be set to the URL in the server url defined in the

Skip to content spec. - If the service account is provided, the service account should grant

`iam.serviceAccounts.getOpenIdToken` permission to Vertex AI Extension Service Agent (<https://cloud.google.com/vertex-ai/docs/general/access-control#service-agents>).

class genai.types.AuthConfigOidcConfigDictBases: `TypedDict`

Config for user OIDC auth.

`id_token`: `Optional[str]`

OpenID Connect formatted ID token for extension endpoint. Only used to propagate token from [[ExecuteExtensionRequest.runtime_auth_config]] at request time.

`service_account`: `Optional[str]`

//cloud.google.com/vertex-ai/docs/general/access-control#service-agents).

TYPE:

The service account used to generate an OpenID Connect (OIDC)-compatible JWT token signed by the Google OIDC Provider (accounts.google.com) for extension endpoint (<https://accounts.google.com>)

TYPE:

//cloud.google.com/iam/docs/create-short-lived-credentials-direct#sa-credentials-oidc). - The audience for the token will be set to the URL in the server url defined in the OpenApi spec. - If the service account is provided, the service account should grant `iam.serviceAccounts.getOpenIdToken` permission to Vertex AI Extension Service Agent (<https://accounts.google.com>)

pydantic model genai.types.AuthTokenBases: `BaseModel`

Config for auth_tokens.create 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:**

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

`field name`: `Optional[str] = None`

Skip to content `e` of the auth token.

class genai.types.AuthTokenDict

Bases: `TypedDict`

Config for `auth_tokens.create` parameters.

`name`: `Optional[str]`

The name of the auth token.

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

Bases: `CaseInsensitiveEnum`

Type of auth scheme.

`API_KEY_AUTH` = `'API_KEY_AUTH'`

API Key Auth.

`AUTH_TYPE_UNSPECIFIED` = `'AUTH_TYPE_UNSPECIFIED'`

`GOOGLE_SERVICE_ACCOUNT_AUTH` = `'GOOGLE_SERVICE_ACCOUNT_AUTH'`

Google Service Account Auth.

`HTTP_BASIC_AUTH` = `'HTTP_BASIC_AUTH'`

HTTP Basic Auth.

`NO_AUTH` = `'NO_AUTH'`

No Auth.

`OAUTHP` = `'OAUTHP'`

OAuth auth.

`OIDC_AUTH` = `'OIDC_AUTH'`

OpenID Connect (OIDC) Auth.

`pydantic model genai.types.AutomaticActivityDetection`

Bases: `BaseModel`

Configures automatic detection of activity.

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.

Only positional-only to allow `self` as a field name.

Skip to content

View schema

FIELDS:

- `disabled (bool | None)`
- `end_of_speech_sensitivity (genai.types.EndSensitivity | None)`
- `prefix_padding_ms (int | None)`
- `silence_duration_ms (int | None)`
- `start_of_speech_sensitivity (genai.types.StartSensitivity | None)`

field disabled: `Optional[bool] = None`

If enabled, detected voice and text input count as activity. If disabled, the client must send activity signals.

field end_of_speech_sensitivity: `Optional[EndSensitivity] = None (alias 'endOfSpeechSensitivity')`

Determines how likely detected speech is ended.

field prefix_padding_ms: `Optional[int] = None (alias 'prefixPaddingMs')`

The required duration of detected speech before start-of-speech is committed. The lower this value the more sensitive the start-of-speech detection is and the shorter speech can be recognized. However, this also increases the probability of false positives.

field silence_duration_ms: `Optional[int] = None (alias 'silenceDurationMs')`

The required duration of detected non-speech (e.g. silence) before end-of-speech is committed. The larger this value, the longer speech gaps can be without interrupting the user's activity but this will increase the model's latency.

field start_of_speech_sensitivity: `Optional[StartSensitivity] = None (alias 'startOfSpeechSensitivity')`

Determines how likely speech is to be detected.

class genai.types.AutomaticActivityDetectionDict

Skip to content

Bases: `TypedDict`

Configures automatic detection of activity.

disabled: `Optional[bool]`

If enabled, detected voice and text input count as activity. If disabled, the client must send activity signals.

end_of_speech_sensitivity: `Optional[EndSensitivity]`

Determines how likely detected speech is ended.

prefix_padding_ms: `Optional[int]`

The required duration of detected speech before start-of-speech is committed. The lower this value the more sensitive the start-of-speech detection is and the shorter speech can be recognized. However, this also increases the probability of false positives.

silence_duration_ms: `Optional[int]`

The required duration of detected non-speech (e.g. silence) before end-of-speech is committed. The larger this value, the longer speech gaps can be without interrupting the user's activity but this will increase the model's latency.

start_of_speech_sensitivity: `Optional[StartSensitivity]`

Determines how likely speech is to be detected.

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

Skip to content

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.

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 `odel`

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')`

The time when the BatchJob was created.

field `dest`: `Optional[BatchJobDestination] = None`

Configuration for the output data.

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

The display name of the BatchJob.

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

The time when the BatchJob was completed.

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`

Skip to content → e of the model that produces the predictions via the BatchJob.

field `name`: `Optional[str] = None`

The resource name of the BatchJob. Output only.”.

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`

The state of the BatchJob.

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

The time when the BatchJob was last updated.

pydantic model genai.types.BatchJobDestination

Skip to content

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:

- `bigrquery_uri` (`str | None`)
- `file_name` (`str | None`)
- `format` (`str | None`)
- `gcs_uri` (`str | None`)
- `inlined_responses` (`list[genai.types.InlinedResponse] | None`)

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

The BigQuery URI to the output table.

field `file_name`: `Optional[str] = None (alias 'fileName')`

The Gemini Developer API's file resource name of the output data (e.g. "files/12345"). The file will be a JSONL file with a single response per line. The responses will be

`GenerateContentResponse` messages formatted as JSON. The responses will be written in the same order as the input requests.

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.

field `inlined_responses`: `Optional[list[InlinedResponse]] = None (alias 'inlinedResponses')`

The responses to the requests in the batch. Returned when the batch was built using inlined requests. The responses will be in the same order as the input requests.

class `genai.types.BatchJobDestinationDict`

Skip to content `Dict`

Config for *des* parameter.

bigquery_uri: `Optional[str]`

The BigQuery URI to the output table.

file_name: `Optional[str]`

The Gemini Developer API's file resource name of the output data (e.g. "files/12345"). The file will be a JSONL file with a single response per line. The responses will be `GenerateContentResponse` messages formatted as JSON. The responses will be written in the same order as the input requests.

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.

inlined_responses: `Optional[list[InlinedResponseDict]]`

The responses to the requests in the batch. Returned when the batch was built using inlined requests. The responses will be in the same order as the input requests.

class `genai.types.BatchJobDict`

Skip to content

Bases: `TypedDict`

Config for batches.create return value.

create_time: `Optional[datetime]`

The time when the BatchJob was created.

dest: `Optional[BatchJobDestinationDict]`

Configuration for the output data.

display_name: `Optional[str]`

The display name of the BatchJob.

end_time: `Optional[datetime]`

The time when the BatchJob was completed.

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]`

The resource name of the BatchJob. Output only.”.

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]`

The state of the BatchJob.

update_time: `Optional[datetime]`

The time when the BatchJob was last updated.

pydantic model `genai.types.BatchJobSource`

Bases: `BaseModel`

Skip to content `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)
- `file_name` (str | None)
- `format` (str | None)
- `gcs_uri` (list[str] | None)
- `inlined_requests` (list[genai.types.InlinedRequest] | None)

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

The BigQuery URI to input table.

field `file_name`: Optional[str] = None (alias 'fileName')

The Gemini Developer API's file resource name of the input data (e.g. "files/12345").

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 URLs to input files.

field `inlined_requests`: Optional[list[InlinedRequest]] = None (alias 'inlinedRequests')

The Gemini Developer API's inlined input data to run batch job.

class `genai.types.BatchJobSourceDict`

Skip to content

Bases: `TypedDict`

Config for `src` parameter.

bigquery_uri: `Optional[str]`

The BigQuery URI to input table.

file_name: `Optional[str]`

The Gemini Developer API's file resource name of the input data (e.g. "files/12345").

format: `Optional[str]`

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

gcs_uris: `Optional[list[str]]`

The Google Cloud Storage URIs to input files.

inlined_requests: `Optional[list[InlinedRequestDict]]`

The Gemini Developer API's inlined input data to run batch job.

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

Bases: `CaseInsensitiveEnum`

Defines the function behavior. Defaults to `BLOCKING`.

BLOCKING = 'BLOCKING'

If set, the system will wait to receive the function response before continuing the conversation.

NON_BLOCKING = 'NON_BLOCKING'

If set, the system will not wait to receive the function response. Instead, it will attempt to handle function responses as they become available while maintaining the conversation between the user and the model.

UNSPECIFIED = 'UNSPECIFIED'

This value is unused.

pydantic model `genai.types.Blob`

Bases: `BaseModel`

Content blob.

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:

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

field `data`: Optional[bytes] = None

Required. Raw bytes.

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

Optional. Display name of the blob. Used to provide a label or filename to distinguish blobs.

This field is not currently used in the Gemini GenerateContent calls.

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.

`display_name`: Optional[str]

Optional. Display name of the blob. Used to provide a label or filename to distinguish blobs.

This field is not currently used in the Gemini GenerateContent calls.

`mime_type`: Optional[str]

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

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

Skip to content

Bases: `CaseInsensitiveEnum`

Output only. Blocked reason.

BLOCKED_REASON_UNSPECIFIED = 'BLOCKED_REASON_UNSPECIFIED'

Unspecified blocked reason.

BLOCKLIST = 'BLOCKLIST'

Candidates blocked due to the terms which are included from the terminology blocklist.

IMAGE_SAFETY = 'IMAGE_SAFETY'

Candidates blocked due to unsafe image generation content.

OTHER = 'OTHER'

Candidates blocked due to other reason.

PROHIBITED_CONTENT = 'PROHIBITED_CONTENT'

Candidates blocked due to prohibited content.

SAFETY = 'SAFETY'

Candidates blocked due to 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`

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

Skip to content

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`

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]`

Skip to content nber 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

Bases: `BaseModel`

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

Skip to content

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)
- url_context_metadata (genai.types.UrlContextMetadata | 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')

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.

-obs_result: Optional[LogprobsResult] = None (alias 'logprobsResult')

Skip to content 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.

field url_context_metadata: `Optional[UrlContextMetadata] = None (alias 'urlContextMetadata')`

Metadata related to url context retrieval tool.

class genai.types.CandidateDict

[Skip to content](#)

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]`

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.

url_context_metadata: `Optional [UrlContextMetadataDict]`

Metadata related to url context retrieval tool.

pydantic model `genai.types.Checkpoint`

Skip to content `odel`

Describes the machine learning model version checkpoint.

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:

- `checkpoint_id` (str | None)
- `epoch` (int | None)
- `step` (int | None)

field `checkpoint_id`: Optional[str] = None (alias 'checkpointId')

The ID of the checkpoint.

field `epoch`: Optional[int] = None

The epoch of the checkpoint.

field `step`: Optional[int] = None

The step of the checkpoint.

class genai.types.CheckpointDict

Bases: TypedDict

Describes the machine learning model version checkpoint.

`checkpoint_id`: Optional[str]

The ID of the checkpoint.

`epoch`: Optional[int]

The epoch of the checkpoint.

`step`: Optional[int]

The step of the checkpoint.

pydantic model genai.types.Citation

Bases: BaseModel

Source attributions for content.

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:

- `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.

Skip to content

`genai.types.CitationMetadataDict`

Bases: `TypedDict`

Citation information when the model quotes another source.

citations: Optional [list [CitationDict]]

Contains 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].

Only generated when using the [CodeExecution] tool, and 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

Skip to content

Bases: `TypedDict`

Result of executing the `[ExecutableCode]`.

Only generated when using the `[CodeExecution]` tool, and 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`

Skip to content `odel`

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.

pydantic model `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

Skip to content

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 empty. 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 empty. 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.

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

► Show JSON schema

Skip to content

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] = None`

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 maximum input.

Skip to content

class genai.types.ContentEmbeddingStatisticsDict

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.ContextWindowCompressionConfig`

Bases: `BaseModel`

Enables context window compression – mechanism managing model context window so it does not exceed given length.

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:

- `sliding_window` (`genai.types.SlidingWindow | None`)
- `trigger_tokens` (`int | None`)

field `sliding_window`: `Optional [SlidingWindow] = None (alias 'slidingWindow')`

Sliding window compression mechanism.

field `trigger_tokens`: `Optional [int] = None (alias 'triggerTokens')`

Number of tokens (before running turn) that triggers context window compression mechanism.

`class genai.types.ContextWindowCompressionConfigDict`

Skip to content

Bases: `TypedDict`

Enables context window compression – mechanism managing model context window so it does not exceed given length.

sliding_window: `Optional[SlidingWindowDict]`

Sliding window compression mechanism.

trigger_tokens: `Optional[int]`

Number of tokens (before running turn) that triggers context window compression mechanism.

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`

Skip to content

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.

pydantic model `genai.types.ControlReferenceImage`

[Skip to content](#)

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`

Skip to content **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.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)`

Skip to content `nSensitiveEnum`

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

Bases: `BaseModel`

Config for the 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.

`Skip to content : 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.CountTokensConfigDictBases: `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.CountTokensResponseBases: `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).

Skip to content `_tokens: Optional[int] = None (alias 'totalTokens')`
total number of tokens.

class genai.types.CountTokensResponseDict

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.CreateAuthTokenConfig

[Skip to content](#)

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:

- `expire_time` (`datetime.datetime | None`)
- `http_options` (`genai.types.HttpOptions | None`)
- `live_connect_constraints` (`genai.types.LiveConnectConstraints | None`)
- `lock_additional_fields` (`list[str] | None`)
- `new_session_expire_time` (`datetime.datetime | None`)
- `uses` (`int | None`)

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

An optional time after which, when using the resulting token, messages in Live API sessions will be rejected. (Gemini may preemptively close the session after this time.)

If not set then this defaults to 30 minutes in the future. If set, this value must be less than 20 hours in the future.

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

Used to override HTTP request options.

field `live_connect_constraints`: `Optional[LiveConnectConstraints] = None (alias 'liveConnectConstraints')`

Configuration specific to Live API connections created using this token.

field `lock_additional_fields`: `Optional[list[str]] = None (alias 'lockAdditionalFields')`

Additional fields to lock in the effective LiveConnectParameters.

field `new_session_expire_time`: `Optional[datetime] = None (alias 'newSessionExpireTime')`

Skip to content after which new Live API sessions using the token resulting from this request will be rejected.

If not set this defaults to 60 seconds in the future. If set, this value must be less than 20 hours in the future.

field `uses`: Optional[int] = None

The number of times the token can be used. If this value is zero then no limit is applied. Default is 1. Resuming a Live API session does not count as a use.

class `genai.types.CreateAuthTokenConfigDict`

Bases: `TypedDict`

Optional parameters.

`expire_time`: Optional[datetime]

An optional time after which, when using the resulting token, messages in Live API sessions will be rejected. (Gemini may preemptively close the session after this time.)

If not set then this defaults to 30 minutes in the future. If set, this value must be less than 20 hours in the future.

`http_options`: Optional[`HttpOptionsDict`]

Used to override HTTP request options.

`live_connect_constraints`: Optional[`LiveConnectConstraintsDict`]

Configuration specific to Live API connections created using this token.

`lock_additional_fields`: Optional[list[str]]

Additional fields to lock in the effective LiveConnectParameters.

`new_session_expire_time`: Optional[datetime]

The time after which new Live API sessions using the token resulting from this request will be rejected.

If not set this defaults to 60 seconds in the future. If set, this value must be less than 20 hours in the future.

`uses`: Optional[int]

The number of times the token can be used. If this value is zero then no limit is applied. Default is 1. Resuming a Live API session does not count as a use.

pydantic model `genai.types.CreateAuthTokenParameters`

Bases: `BaseModel`

Skip to content

`tokens.create` 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:

- config (genai.types.CreateAuthTokenConfig | None)

field config: Optional [CreateAuthTokenConfig] = None

Optional parameters for the request.

class genai.types.CreateAuthTokenParametersDict

Bases: TypedDict

Config for auth_tokens.create parameters.

config: Optional [CreateAuthTokenConfigDict]

Optional parameters for the request.

pydantic model genai.types.CreateBatchJobConfig

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:

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

field `dest`: `Union[BatchJobDestination, str, None] = 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`

Bases: `TypedDict`

Config for optional parameters.

`dest`: `Union[BatchJobDestination, str, BatchJobDestinationDict, None]`

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.

Skip to content

pydantic_model `genai.types.CreateCachedContentConfig`

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)
- `kms_key_name` (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)
- `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. Uses RFC 3339 format, Example: 2014-10-02T15:01:23Z.

Skip to content **options:** Optional[HttpOptions] = None (alias 'httpOptions')

Used to override HTTP request options.

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

The Cloud KMS resource identifier of the customer managed encryption key used to protect a resource. The key needs to be in the same region as where the compute resource is created. See <https://cloud.google.com/vertex-ai/docs/general/cmek> for more details. If this is set, then all created CachedContent objects will be encrypted with the provided encryption key. Allowed formats:

projects/{project}/locations/{location}/keyRings/{key_ring}/cryptoKeys/{crypto_key}

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

Developer set system instruction.

```
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. It is a duration string, with up to nine fractional digits, terminated by 's'. Example: "3.5s".

```
class genai.types.CreateCachedContentConfigDict
```

Skip to content

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]`

01:23Z.

TYPE:

Timestamp of when this resource is considered expired. Uses RFC 3339 format,

Example

TYPE:

2014-10-02T15

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

kms_key_name: `Optional[str]`

The Cloud KMS resource identifier of the customer managed encryption key used to protect a resource. The key needs to be in the same region as where the compute resource is created. See <https://cloud.google.com/vertex-ai/docs/general/cmek> for more details. If this is set, then all created CachedContent objects will be encrypted with the provided encryption key. Allowed formats:

`projects/{project}/locations/{location}/keyRings/{key_ring}/cryptoKeys/{crypto_key}`

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]]`

Skip to content `Tools` the model may use to generate the next response.

ttl: `Optional[str]`

“3.5s”.

TYPE:

The TTL for this resource. The expiration time is computed

TYPE:

now + TTL. It is a duration string, with up to nine fractional digits, terminated by ‘s’.

Example

`pydantic model genai.types.CreateFileConfig`

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`)
- `should_return_http_response` (`bool | None`)

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

Used to override HTTP request options.

`field should_return_http_response: Optional[bool] = None (alias 'shouldReturnHttpResponse')`

If true, the raw HTTP response will be returned in the ‘`sdk_http_response`’ field.

`class genai.types.CreateFileConfigDict`

Bases: `TypedDict`

Used to override the default configuration.

`http_options: Optional[HttpOptionsDict]`

Used to override HTTP request options.

`should_return_http_response: Optional[bool]`

Skip to content the raw HTTP response will be returned in the ‘`sdk_http_response`’ field.

`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:

- `sdk_http_response` (`genai.types.HttpResponse` | `None`)

field `sdk_http_response`: `Optional[HttpResponse]` = `None` (alias '`sdkHttpResponse`')

Used to retain the full HTTP response.

class `genai.types.CreateFileResponseDict`

Bases: `TypedDict`

Response for the create file method.

`sdk_http_response`: `Optional[HttpResponseDict]`

Used to retain the full HTTP response.

pydantic model `genai.types.CreateTuningJobConfig`

Bases: `BaseModel`

Supervised fine-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

Skip to content

FIELDS:

- `adapter_size (genai.types.AdapterSize | None)`
- `batch_size (int | None)`
- `description (str | None)`
- `epoch_count (int | None)`
- `export_last_checkpoint_only (bool | 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 `export_last_checkpoint_only`: `Optional[bool] = None (alias 'exportLastCheckpointOnly')`

If set to true, disable intermediate checkpoints for SFT and only the last checkpoint will be exported. Otherwise, enable intermediate checkpoints for SFT.

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.

Skip to content **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' )
```

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

```
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
```

[Skip to content](#)

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.

export_last_checkpoint_only: `Optional[bool]`

If set to true, disable intermediate checkpoints for SFT and only the last checkpoint will be exported. Otherwise, enable intermediate checkpoints for SFT.

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.

Skip to content [genai.types.DatasetDistribution](#)

Bases: `BaseModel`

Distribution computed over a tuning 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:

- `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.

Skip to content **Optional [float] = None**

Output only. Sum of a given population of values.

class genai.types.DatasetDistributionDictBases: `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.DatasetDistributionDistributionBucketBases: `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.

.. ^__^ N schema

Skip to content

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.DatasetDistributionDistributionBucketDictBases: `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.DatasetStatsBases: `BaseModel`

Statistics computed over a tuning 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

Skip to content

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')`

Skip to content only. Dataset distributions for the user output tokens.

class `genai.types.DatasetStatsDict`

Bases: `TypedDict`

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

Skip to content

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`

Bases: `TypedDict`

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.

Skip to content `genai.types.DeleteCachedContentResponse`

`model`

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

Empty response 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

Skip to content

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

Configuration for deleting a tuned 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:

- 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

Configuration for deleting a tuned model.

http_options: Optional[HttpOptionsDict]

Used to override HTTP request options.

pydantic model genai.types.DeleteModelResponse

Skip to 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

`class genai.types.DeleteModelResponseDict`

Bases: `TypedDict`

`pydantic model genai.types.DeleteResourceJob`

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]`

Skip to content

`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.

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:

- `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
```

Skip to content

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.

class `genai.types.DistillationHyperParametersDict`

Bases: `TypedDict`

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]`

Skip to content Multiplier for adjusting the default learning rate.

pydantic model `genai.types.DistillationSpec`

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. See [Supported models]
(https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/tuning#supported_models).

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

Optional. Hyperparameters for Distillation.

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

Deprecated. 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”. Deprecated. Use `base_model` instead.

Skip to content **field `training_dataset_uri`:** `Optional[str] = None (alias 'trainingDatasetUri')`

Deprecated. 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}.

```
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.DistillationSpecDict
```

[Skip to content](#)

Bases: `TypedDict`

Tuning Spec for Distillation.

base_teacher_model: `Optional [str]`

//cloud.google.com/vertex-ai/generative-ai/docs/model-reference/tuning#supported_models).

TYPE:

The base teacher model that is being distilled. See [Supported models](https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/tuning#supported_models).

hyper_parameters: `Optional [DistillationHyperParametersDict]`

Optional. Hyperparameters for Distillation.

pipeline_root_directory: `Optional [str]`

Deprecated. 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”. Deprecated. Use `base_model` instead.

training_dataset_uri: `Optional [str]`

Deprecated. 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.

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.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.

Skip to content **`Optional[DynamicRetrievalConfigMode]`** = `None`

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

class genai.types.DynamicRetrievalConfigDictBases: `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)Bases: `CaseInSensitiveEnum`

Config for the dynamic retrieval config mode.

MODE_DYNAMIC = 'MODE_DYNAMIC'

Run retrieval only when system decides it is necessary.

MODE_UNSPECIFIED = 'MODE_UNSPECIFIED'

Always trigger retrieval.

pydantic model genai.types.EditImageConfigBases: `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

Skip to content

FIELDS:

- `add_watermark (bool | None)`
- `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 `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. Supported values are “1:1”, “3:4”, “4:3”, “9:16”, and “16:9”.

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 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 generated image and the positive prompt 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.

Skip to content [yes](#).[EditImageConfigDict](#)

Bases: `TypedDict`

Configuration for editing an image.

add_watermark: `Optional[bool]`

Whether to add a watermark to the generated images.

aspect_ratio: `Optional[str]`

Aspect ratio of the generated images. Supported values are “1:1”, “3:4”, “4:3”, “9:16”, and “16:9”.

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 generated image and the positive prompt 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.

Skip to content **resession_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]

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.

Skip to content `mages:` 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'
```

```
EDIT_MODE_PRODUCT_IMAGE = 'EDIT_MODE_PRODUCT_IMAGE'
```

```
EDIT_MODE_STYLE = 'EDIT_MODE_STYLE'
```

```
pydantic model genai.types.EmbedContentConfig
```

[Skip to content](#)

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 `skip_text`: `Optional[str] = None`

Skip to content in the text. Only applicable when TaskType is `RETRIEVAL_DOCUMENT`.

class `genai.types.EmbedContentConfigDict`

Bases: `TypedDict`

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)`

Skip to content

ble_character_count: `Optional[int] = None` (alias
`'billableCharacterCount'`)

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

class genai.types.EmbedContentMetadataDictBases: `TypedDict`

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.EmbedContentResponseBases: `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.EmbedContentResponseDictBases: `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.

Skip to content

genai.types.EncryptionSpecBases: `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 form a valid 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

class genai.types.EndSensitivity(*values)

Skip to content

Bases: CaseInsensitiveEnum

End of speech sensitivity.

END_SENSITIVITY_HIGH = 'END_SENSITIVITY_HIGH'

Automatic detection ends speech more often.

END_SENSITIVITY_LOW = 'END_SENSITIVITY_LOW'

Automatic detection ends speech less often.

END_SENSITIVITY_UNSPECIFIED = 'END_SENSITIVITY_UNSPECIFIED'

The default is END_SENSITIVITY_LOW.

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.

class genai.types.EndpointDict

Skip to content

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.EnterpriseWebSearch

Bases: `BaseModel`

Tool to search public web data, powered by Vertex AI Search and Sec4 compliance.

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.EnterpriseWebSearchDict

Bases: `TypedDict`

Tool to search public web data, powered by Vertex AI Search and Sec4 compliance.

class genai.types.Environment(*values)

Bases: `CaseInsensitiveEnum`

Required. The environment being operated.

ENVIRONMENT_BROWSER = 'ENVIRONMENT_BROWSER'

Operates in a web browser.

ENVIRONMENT_UNSPECIFIED = 'ENVIRONMENT_UNSPECIFIED'

Defaults to browser.

pydantic model genai.types.ExecutableCode

Bases: `BaseModel`

Skip to content generated by the model that is meant to be executed, and the result returned to the user.

Generated when using the [CodeExecution] tool, in which the code will be automatically executed, and a corresponding [CodeExecutionResult] will also be generated.

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 [CodeExecution] tool, in which the code will be automatically executed, and a corresponding [CodeExecutionResult] will also be generated.

code: `Optional[str]`

Required. The code to be executed.

language: `Optional[Language]`

Required. Programming language of the code.

pydantic model genai.types.ExternalApi

Bases: `BaseModel`

Retrieve from data source powered by external API for grounding.

API is not owned by Google, but need to follow the pre-defined API spec.

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:

- `api_auth` (`genai.types.ApiAuth | None`)
- `api_spec` (`genai.types.ApiSpec | None`)
- `auth_config` (`genai.types.AuthConfig | None`)
- `elastic_search_params` (`genai.types.ExternalApiElasticSearchParams | None`)
- `endpoint` (`str | None`)
- `simple_search_params` (`genai.types.ExternalApiSimpleSearchParams | None`)

field `api_auth`: `Optional[ApiAuth] = None (alias 'apiAuth')`

The authentication config to access the API. Deprecated. Please use `auth_config` instead.

field `api_spec`: `Optional[ApiSpec] = None (alias 'apiSpec')`

The API spec that the external API implements.

field `auth_config`: `Optional[AuthConfig] = None (alias 'authConfig')`

The authentication config to access the API.

field `elastic_search_params`: `Optional[ExternalApiElasticSearchParams] = None (alias 'elasticSearchParams')`

Parameters for the elastic search API.

field `endpoint`: `Optional[str] = None`

The endpoint of the external API. The system will call the API at this endpoint to retrieve the data for grounding. Example: <https://acme.com:443/search>

field `simple_search_params`: `Optional[ExternalApiSimpleSearchParams] = None (alias 'simpleSearchParams')`

Parameters for the simple search API.

class `genai.types.ExternalApiDict`

Bases: `TypedDict`

Retrieve from data source powered by external API for grounding.

Skip to content

The external API is not owned by Google, but need to follow the pre-defined API spec.

api_auth: Optional [ApiAuthDict]

The authentication config to access the API. Deprecated. Please use auth_config instead.

api_spec: Optional [ApiSpec]

The API spec that the external API implements.

auth_config: Optional [AuthConfigDict]

The authentication config to access the API.

elastic_search_params: Optional [ExternalApiElasticSearchParamsDict]

Parameters for the elastic search API.

endpoint: Optional [str]

//acme.com:443/search

TYPE:

The endpoint of the external API. The system will call the API at this endpoint to retrieve the data for grounding. Example

TYPE:

https

simple_search_params: Optional [ExternalApiSimpleSearchParamsDict]

Parameters for the simple search API.

pydantic model genai.types.ExternalApiElasticSearchParams

Bases: BaseModel

The search parameters to use for the ELASTIC_SEARCH spec.

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:

- `index (str | None)`
- `num_hits (int | None)`
- `search_template (str | None)`

field index: `Optional[str] = None`

The ElasticSearch index to use.

field num_hits: `Optional[int] = None (alias 'numHits')`

Optional. Number of hits (chunks) to request. When specified, it is passed to Elasticsearch as the `num_hits` param.

field search_template: `Optional[str] = None (alias 'searchTemplate')`

The ElasticSearch search template to use.

class genai.types.ExternalApiElasticSearchParamsDict

Bases: `TypedDict`

The search parameters to use for the ELASTIC_SEARCH spec.

index: `Optional[str]`

The ElasticSearch index to use.

num_hits: `Optional[int]`

Optional. Number of hits (chunks) to request. When specified, it is passed to Elasticsearch as the `num_hits` param.

search_template: `Optional[str]`

The ElasticSearch search template to use.

pydantic model genai.types.ExternalApiSimpleSearchParams

Bases: `BaseModel`

The search parameters to use for SIMPLE_SEARCH spec.

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 implicitly positional-only to allow `self` as a field name.

Skip to content

View schema

class genai.types.ExternalApiSimpleSearchParamsDict

Bases: `TypedDict`

The search parameters to use for SIMPLE_SEARCH spec.

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

Bases: `CaseInsensitiveEnum`

Options for feature selection preference.

`BALANCED` = 'BALANCED'

`FEATURE_SELECTION_PREFERENCE_UNSPECIFIED` =
'FEATURE_SELECTION_PREFERENCE_UNSPECIFIED'

`PRIORITIZE_COST` = 'PRIORITIZE_COST'

`PRIORITIZE_QUALITY` = 'PRIORITIZE_QUALITY'

```
pydantic model 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`

Skip to content and 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

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 be deleted.

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*

field sha256_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.

Skip to content \ schema

FIELDS:

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

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

Optional. Display name of the file data. Used to provide a label or filename to distinguish file datas. It is not currently used in the Gemini GenerateContent calls.

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`

Bases: `TypedDict`

URI based data.

`display_name`: `Optional[str]`

Optional. Display name of the file data. Used to provide a label or filename to distinguish file datas. It is not currently used in the Gemini GenerateContent calls.

`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]`

Skip to content only. Size of the file in bytes.

source: `Optional[FileSource]`

Output only. The source of the *File*.

state: `Optional[FileState]`

Output only. 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

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.

`-----[ValidationModelError][pydantic_core.ValidationError]` if the input data cannot be validated to Skip to content node.

`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)

Skip to content

Bases: CaseInsensitiveEnum

Output only. The reason why the model stopped generating tokens.

If empty, the model has not stopped generating the tokens.

BLOCKLIST = 'BLOCKLIST'

Token generation stopped because the content contains forbidden terms.

FINISH_REASON_UNSPECIFIED = 'FINISH_REASON_UNSPECIFIED'

The finish reason is unspecified.

IMAGE_SAFETY = 'IMAGE_SAFETY'

Token generation stopped because generated images have safety violations.

LANGUAGE = 'LANGUAGE'

The token generation stopped because of using an unsupported language.

MALFORMED_FUNCTION_CALL = 'MALFORMED_FUNCTION_CALL'

The function call generated by the model is invalid.

MAX_TOKENS = 'MAX_TOKENS'

Token generation reached the configured maximum output tokens.

OTHER = 'OTHER'

All other reasons that stopped the token generation.

PROHIBITED_CONTENT = 'PROHIBITED_CONTENT'

Token generation stopped for potentially containing prohibited content.

RECITATION = 'RECITATION'

The token generation stopped because of potential recitation.

SAFETY = 'SAFETY'

When streaming, [content] is empty if content filters blocks the output.

TYPE:

Token generation stopped because the content potentially contains safety violations.

NOTE

SPII = 'SPII'

Skip to content generation stopped because the content potentially contains Sensitive Personally Identifiable Information (SPII).

STOP = 'STOP'

Token generation reached a natural stopping point or a configured stop sequence.

UNEXPECTED_TOOL_CALL = 'UNEXPECTED_TOOL_CALL'

The tool call generated by the model is invalid.

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. 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]`.

class genai.types.FunctionCallDict

Skip to content

Bases: TypedDict

A function call.

args: Optional[dict[str, Any]]

Optional. 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.

Skip to content [genai.types.FunctionCallingConfigDict](#)

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'

Model is constrained to always predicting function calls only. If "allowed_function_names" are set, the predicted function calls will be limited to any one of "allowed_function_names", else the predicted function calls will be any one of the provided "function_declarations".

AUTO = 'AUTO'

Default model behavior, model decides to predict either function calls or natural language response.

MODE_UNSPECIFIED = 'MODE_UNSPECIFIED'

The function calling config mode is unspecified. Should not be used.

NONE = 'NONE'

Model will not predict any function calls. Model behavior is same as when not passing any function declarations.

pydantic model genai.types.FunctionDeclaration

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 Skip to content node.

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

► Show JSON schema

FIELDS:

- `behavior` (`genai.types.Behavior | None`)
- `description` (`str | None`)
- `name` (`str | None`)
- `parameters` (`genai.types.Schema | None`)
- `parameters_json_schema` (`Any | None`)
- `response` (`genai.types.Schema | None`)
- `response_json_schema` (`Any | None`)

field behavior: `Optional[Behavior] = None`

Defines the function behavior.

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 parameters_json_schema: `Optional[Any] = None (alias 'parametersJsonSchema')`

Optional. Describes the parameters to the function in JSON Schema format. The schema must describe an object where the properties are the parameters to the function. For example: `` { "type": "object", "properties": { "name": { "type": "string" }, "age": { "type": "integer" } }, "additionalProperties": false, "required": ["name", "age"], "propertyOrdering": ["name", "age"] }`` This field is mutually exclusive with `parameters`.

Skip to content `"age"]`, `"propertyOrdering": ["name", "age"] }` This field is mutually exclusive with `parameters`.

```
field response: Optional[Schema] = None
```

Optional. Describes the output from this function in JSON Schema format. Reflects the Open API 3.03 Response Object. The Schema defines the type used for the response value of the function.

```
field response_json_schema: Optional[Any] = None (alias 'responseJsonSchema')
```

Optional. Describes the output from this function in JSON Schema format. The value specified by the schema is the response value of the function. This field is mutually exclusive with *response*.

```
classmethod from_callable(*, client, callable, behavior=None)
```

Converts a Callable to a FunctionDeclaration based on the client.

RETURN TYPE:

```
FunctionDeclaration
```

```
classmethod from_callable_with_api_option(*, callable, api_option='GEMINI_API', behavior=None)
```

Converts a Callable to a FunctionDeclaration based on the API option.

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
```

[Skip to content](#)

Bases: `TypedDict`

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

The inputs are based on OpenAPI 3.0 specifications.

behavior: `Optional[Behavior]`

Defines the function behavior.

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

parameters_json_schema: `Optional[Any]`

```
` { "type": "object", "properties": { "name": { "type": "string" }, "age": { "type": "integer" } }, "additionalProperties": false, "required": [ "name", "age" ], "propertyOrdering": [ "name", "age" ] }` This field is mutually exclusive with parameters.
```

TYPE:

Optional. Describes the parameters to the function in JSON Schema format. The schema must describe an object where the properties are the parameters to the function. For example

Skip to content

`Optional[SchemaDict]`

Optional. Describes the output from this function in JSON Schema format. Reflects the Open API 3.03 Response Object. The Schema defines the type used for the response value of the function.

response_json_schema: `Optional[Any]`

Optional. Describes the output from this function in JSON Schema format. The value specified by the schema is the response value of the function. This field is mutually exclusive with *response*.

pydantic model genai.types.FunctionResponse

[Skip to content](#)

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.

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

► Show JSON schema

FIELDS:

- `id (str | None)`
- `name (str | None)`
- `response (dict[str, Any] | None)`
- `scheduling (genai.types.FunctionResponseScheduling | None)`
- `will_continue (bool | None)`

field `id`: `Optional[str] = None`

Optional. 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.

field `scheduling`: `Optional[FunctionResponseScheduling] = None`

Specifies how the response should be scheduled in the conversation. Only applicable to NON_BLOCKING function calls, is ignored otherwise. Defaults to WHEN_IDLE.

field `will_continue`: `Optional[bool] = None (alias 'willContinue')`

Signals that function call continues, and more responses will be returned, turning the function call into a generator. Is only applicable to NON_BLOCKING function calls (see

`Declaration.behavior` for details), ignored otherwise. If false, the default, future

Skip to content `is` will not be considered. Is only applicable to NON_BLOCKING function calls, is ignored otherwise. If set to false, future responses will not be considered. It is allowed to return empty `response` with `will_continue=False` to signal that the function call is finished.

```
classmethod from_mcp_response(*, name, response)
```

RETURN TYPE:

FunctionResponse

```
class genai.types.FunctionResponseDict
```

Bases: TypedDict

A function response.

id: Optional[str]

Optional. 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.

scheduling: Optional[FunctionResponseScheduling]

Specifies how the response should be scheduled in the conversation. Only applicable to NON_BLOCKING function calls, is ignored otherwise. Defaults to WHEN_IDLE.

will_continue: Optional[bool]

Signals that function call continues, and more responses will be returned, turning the function call into a generator. Is only applicable to NON_BLOCKING function calls (see FunctionDeclaration.behavior for details), ignored otherwise. If false, the default, future responses will not be considered. Is only applicable to NON_BLOCKING function calls, is ignored otherwise. If set to false, future responses will not be considered. It is allowed to return empty *response* with *will_continue=False* to signal that the function call is finished.

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

Skip to content

Bases: `CaseInsensitiveEnum`

Specifies how the response should be scheduled in the conversation.

INTERRUPT = `'INTERRUPT'`

Add the result to the conversation context, interrupt ongoing generation and prompt to generate output.

SCHEDULING_UNSPECIFIED = `'SCHEDULING_UNSPECIFIED'`

This value is unused.

SILENT = `'SILENT'`

Only add the result to the conversation context, do not interrupt or trigger generation.

WHEN_IDLE = `'WHEN_IDLE'`

Add the result to the conversation context, and prompt to generate output without interrupting ongoing generation.

`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.

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

► Show JSON schema

Skip to content

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)`
- `model_selection_config (genai.types.ModelSelectionConfig | None)`
- `presence_penalty (float | None)`
- `response_json_schema (Any | None)`
- `response_logprobs (bool | None)`
- `response_mime_type (str | None)`
- `response_modalities (list[str] | None)`
- `response_schema (dict[Any, Any] | 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[..., Any] | mcp.types.Tool | mcp.client.session.ClientSession] | None)`
 - `'float | None)`

Skip to content `'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.

field `automatic_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 `model_selection_config`: `Optional[ModelSelectionConfig] = None (alias 'modelSelectionConfig')`

Configuration for model selection.

field `presence_penalty`: `Optional[float] = None (alias 'presencePenalty')`

Skip to content values penalize tokens that already appear in the generated text, increasing the probability of generating more diverse content.

```
field response_json_schema: Optional[Any] = None (alias 'responseJsonSchema')
```

Optional. Output schema of the generated response. This is an alternative to `response_schema` that accepts [JSON Schema](<https://json-schema.org/>). If set, `response_schema` must be omitted, but `response_mime_type` is required. While the full JSON Schema may be sent, not all features are supported. Specifically, only the following properties are supported: - `$id` - `$defs` - `$ref` - `$anchor` - `type` - `format` - `title` - `description` - `enum` (for strings and numbers) - `items` - `prefixItems` - `minItems` - `maxItems` - `minimum` - `maximum` - `anyOf` - `oneOf` (interpreted the same as `anyOf`) - `properties` - `additionalProperties` - `required` The non-standard `propertyOrdering` property may also be set. Cyclic references are unrolled to a limited degree and, as such, may only be used within non-required properties. (Nullable properties are not sufficient.) If `$ref` is set on a sub-schema, no other properties, except for those starting as a \$, may be set.

```
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 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_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[Any, Any], type, Schema, GenericAlias, , _UnionGenericAlias, None] = None (alias 'responseSchema')
```

Skip to content

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.

VALIDATED BY:

- `_convert_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.

Skip to content **config:** `Optional[ToolConfig] = None (alias 'toolConfig')`

Associates model output to a specific function call.

```
field tools: Optional[ list[ Union[ Tool, Callable[ ..., Any ], Tool, ClientSession ] ] ] = 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 random responses.

```
class genai.types.GenerateContentConfigDict
```

Skip to content

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.

model_selection_config: `Optional[ModelSelectionConfigDict]`

Configuration for model selection.

presence_penalty: `Optional[float]`

Skip to content values penalize tokens that already appear in the generated text, increasing the probability of generating more diverse content.

response_json_schema: Optional [Any]

Optional. Output schema of the generated response. This is an alternative to `response_schema` that accepts [JSON Schema](<https://json-schema.org/>). If set, `response_schema` must be omitted, but `response_mime_type` is required. While the full JSON Schema may be sent, not all features are supported. Specifically, only the following properties are supported: - `$id` - `$defs` - `$ref` - `$anchor` - `type` - `format` - `title` - `description` - `enum` (for strings and numbers) - `items` - `prefixItems` - `minItems` - `maxItems` - `minimum` - `maximum` - `anyOf` - `oneOf` (interpreted the same as `anyOf`) - `properties` - `additionalProperties` - `required` The non-standard `propertyOrdering` property may also be set. Cyclic references are unrolled to a limited degree and, as such, may only be used within non-required properties. (Nullable properties are not sufficient.) If `$ref` is set on a sub-schema, no other properties, except for those starting as a \$, may be set.

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]

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.

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 [Any , Any], type , Schema , GenericAlias , , _UnionGenericAlias , SchemaDict , None]

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

I response.

Skip to content

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 [... , Any] , Tool , ClientSession]]]

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]

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.

Skip to content

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[Any, Any] | enum.Enum | None`)
- `prompt_feedback` (`genai.types.GenerateContentResponsePromptFeedback | None`)
- `response_id` (`str | None`)
- `sdk_http_response` (`genai.types.HttpResponse | 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[Any, Any], Enum, None] = None`

First candidate from the parsed response if `response_schema` is provided. Not available for streaming.

field `prompt_feedback`: `Optional[GenerateContentResponsePromptFeedback] = None (alias 'Feedback')`

Skip to content

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 sdk_http_response: `Optional[HttpResponse] = None (alias 'sdkHttpResponse')`

Used to retain the full HTTP 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

Skip to content

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.

sdk_http_response: `Optional [HttpResponseDict]`

Used to retain the full HTTP 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.

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:

- `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:

- `cache_tokens_details` (`list[genai.types.ModalityTokenCount] | None`)
- `cached_content_token_count` (`int | None`)
- `candidates_token_count` (`int | None`)
- `candidates_tokens_details` (`list[genai.types.ModalityTokenCount] | None`)
- `prompt_token_count` (`int | None`)
- `prompt_tokens_details` (`list[genai.types.ModalityTokenCount] | None`)
- `thoughts_token_count` (`int | None`)
- `tool_use_prompt_token_count` (`int | None`)
- `tool_use_prompt_tokens_details` (`list[genai.types.ModalityTokenCount] | None`)
- `total_token_count` (`int | None`)
- `traffic_type` (`genai.types.TrafficType | None`)

field `cache_tokens_details`: `Optional[list[ModalityTokenCount]] = None` (alias '`cacheTokensDetails`')

Output only. List of modalities of the cached content in the request input.

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 `candidates_tokens_details`: `Optional[list[ModalityTokenCount]] = None` (alias '`candidatesTokensDetails`')

Output only. List of modalities that were returned in the response.

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 `prompt_tokens_details`: `Optional[list[ModalityTokenCount]] = None` (alias '`promptTokensDetails`')

Output only. List of modalities that were processed in the request input.

field `qhts_token_count`: `Optional[int] = None` (alias '`thoughtsTokenCount`')

Skip to content only. Number of tokens present in thoughts output.

field `tool_use_prompt_token_count`: `Optional[int] = None` (alias '`toolUsePromptTokenCount`')

```
'toolUsePromptTokenCount')
```

Output only. Number of tokens present in tool-use prompt(s).

```
field tool_use_prompt_tokens_details: Optional[list[ModalityTokenCount]] = None  
(alias 'toolUsePromptTokensDetails')
```

Output only. List of modalities that were processed for tool-use request inputs.

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

Total token count for prompt, response candidates, and tool-use prompts (if present).

```
field traffic_type: Optional[TrafficType] = None (alias 'trafficType')
```

Output only. Traffic type. This shows whether a request consumes Pay-As-You-Go or Provisioned Throughput quota.

```
class genai.types.GenerateContentResponseUsageMetadataDict
```

Skip to content

Bases: `TypedDict`

Usage metadata about response(s).

cache_tokens_details: `Optional [list [ModalityTokenCountDict]]`

Output only. List of modalities of the cached content in the request input.

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).

candidates_tokens_details: `Optional [list [ModalityTokenCountDict]]`

Output only. List of modalities that were returned in the response.

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.

prompt_tokens_details: `Optional [list [ModalityTokenCountDict]]`

Output only. List of modalities that were processed in the request input.

thoughts_token_count: `Optional [int]`

Output only. Number of tokens present in thoughts output.

tool_use_prompt_token_count: `Optional [int]`

Output only. Number of tokens present in tool-use prompt(s).

tool_use_prompt_tokens_details: `Optional [list [ModalityTokenCountDict]]`

Output only. List of modalities that were processed for tool-use request inputs.

total_token_count: `Optional [int]`

Total token count for prompt, response candidates, and tool-use prompts (if present).

traffic_type: `Optional [TrafficType]`

Output only. Traffic type. This shows whether a request consumes Pay-As-You-Go or Provisioned Throughput quota.

pydantic model `genai.types.GenerateImagesConfig`

Skip to content `odel`

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. Supported values are "1:1", "3:4", "4:3", "9:16", and "16:9".

field enhance_prompt: Optional[bool] = None (alias 'enhancePrompt')

Whether to use the prompt rewriting logic.

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

Skip to content 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')`

Whether to report the safety scores of each generated image and the positive prompt 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`

Skip to content 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. Supported values are “1:1”, “3:4”, “4:3”, “9:16”, and “16:9”.

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 generated image and the positive prompt 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).

Skip to content

.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]`

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.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`)
- `positive_prompt_safety_attributes` (`genai.types.SafetyAttributes | None`)

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

List of generated images.

field positive_prompt_safety_attributes: `Optional[SafetyAttributes] = None (alias 'positivePromptSafetyAttributes')`

Safety attributes of the positive prompt. Only populated if `include_safety_attributes` is set to True.

class genai.types.GenerateImagesResponseDict

Skip to content `Dict`

The output images response.

generated_images: `Optional [list [GeneratedImageDict]]`

List of generated images.

positive_prompt_safety_attributes: `Optional [SafetyAttributesDict]`

Safety attributes of the positive prompt. Only populated if `include_safety_attributes` is set to True.

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)`
- `compression_quality (genai.types.VideoCompressionQuality | None)`
- `duration_seconds (int | None)`
- `enhance_prompt (bool | None)`
- `fps (int | None)`
- `generate_audio (bool | None)`
- `http_options (genai.types.HttpOptions | None)`
- `last_frame (genai.types.Image | 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 compression_quality: `Optional[VideoCompressionQuality] = None (alias 'compressionQuality')`

Compression quality of the generated videos.

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 generate_audio: `Optional[bool] = None (alias 'generateAudio')`

Whether to generate audio along with the video.

Skip to content **options:** `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

field last_frame: Optional[Image] = None (alias 'lastFrame')

Image to use as the last frame of generated videos. Only supported for image to video use cases.

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.

field pubsub_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. 720p and 1080p 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

compression_quality: `Optional [VideoCompressionQuality]`

Compression quality of the generated videos.

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.

generate_audio: `Optional [bool]`

Whether to generate audio along with the video.

http_options: `Optional [HttpOptionsDict]`

Used to override HTTP request options.

last_frame: `Optional [ImageDict]`

Image to use as the last frame of generated videos. Only supported for image to video use cases.

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.

Skip to content

_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. 720p and 1080p are supported.

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`, `Operation`

A video generation 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:

- `response` (`genai.types.GenerateVideosResponse` | `None`)
- `result` (`genai.types.GenerateVideosResponse` | `None`)

field `response`: `Optional[GenerateVideosResponse] = None`

The generated videos.

field `result`: `Optional[GenerateVideosResponse] = None`

The generated videos.

classmethod `from_api_response`(`api_response`, `is_vertex_ai=False`)

Instantiates a `GenerateVideosOperation` from an API response.

RETURN TYPE:

`Self`

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.

Skip to content ↗ Only positional-only to allow `self` as a field name.

► Show JSON schema

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`

video.

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:

- `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

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

Skip to content

FIELDS:

- `audio_timestamp (bool | None)`
- `candidate_count (int | None)`
- `enable_affective_dialog (bool | None)`
- `frequency_penalty (float | None)`
- `logprobs (int | None)`
- `max_output_tokens (int | None)`
- `media_resolution (genai.types.MediaResolution | None)`
- `model_selection_config (genai.types.ModelSelectionConfig | None)`
- `presence_penalty (float | None)`
- `response_json_schema (Any | None)`
- `response_logprobs (bool | None)`
- `response_mime_type (str | None)`
- `response_modalities (list[genai.types.Modality] | None)`
- `response_schema (genai.types.Schema | None)`
- `routing_config (genai.types.GenerationConfigRoutingConfig | None)`
- `seed (int | None)`
- `speech_config (genai.types.SpeechConfig | None)`
- `stop_sequences (list[str] | None)`
- `temperature (float | None)`
- `thinking_config (genai.types.GenerationConfigThinkingConfig | 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 `enable_affective_dialog`: `Optional[bool] = None (alias 'enableAffectiveDialog')`

Optional. If enabled, the model will detect emotions and adapt its responses accordingly.

field `frequency_penalty`: `Optional[float] = None (alias 'frequencyPenalty')`

Skip to content 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.

```
field media_resolution: Optional[ MediaResolution ] = None (alias 'mediaResolution')
```

Optional. If specified, the media resolution specified will be used.

```
field model_selection_config: Optional[ ModelSelectionConfig ] = None (alias 'modelSelectionConfig')
```

Optional. Config for model selection.

```
field presence_penalty: Optional[ float ] = None (alias 'presencePenalty')
```

Optional. Positive penalties.

```
field response_json_schema: Optional[ Any ] = None (alias 'responseJsonSchema')
```

Optional. Output schema of the generated response. This is an alternative to `response_schema` that accepts [JSON Schema](<https://json-schema.org/>). If set, `response_schema` must be omitted, but `response_mime_type` is required. While the full JSON Schema may be sent, not all features are supported. Specifically, only the following properties are supported: - `$id` - `$defs` - `$ref` - `$anchor` - `type` - `format` - `title` - `description` - `enum` (for strings and numbers) - `items` - `prefixItems` - `minItems` - `maxItems` - `minimum` - `maximum` - `anyOf` - `oneOf` (interpreted the same as `anyOf`) - `properties` - `additionalProperties` - `required` The non-standard `propertyOrdering` property may also be set. Cyclic references are unrolled to a limited degree and, as such, may only be used within non-required properties. (Nullable properties are not sufficient.) If `$ref` is set on a sub-schema, no other properties, except for those starting as a \$, may be set.

```
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_modalities: Optional[ list[ Modality ] ] = None (alias 'responseModalities')
```

^ .. 'The modalities of the response.'

Skip to content

```
. 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 speech_config: Optional[SpeechConfig] = None (alias 'speechConfig')
```

Optional. The speech generation config.

```
field stop_sequences: Optional[list[str]] = None (alias 'stopSequences')
```

Optional. Stop sequences.

```
field temperature: Optional[float] = None
```

Optional. Controls the randomness of predictions.

```
field thinking_config: Optional[GenerationConfigThinkingConfig] = None (alias  
    'thinkingConfig')
```

Optional. Config for thinking features. An error will be returned if this field is set for models that don't support thinking.

```
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
```

Skip to content

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.

enable_affective_dialog: `Optional[bool]`

Optional. If enabled, the model will detect emotions and adapt its responses accordingly.

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.

media_resolution: `Optional[MediaResolution]`

Optional. If specified, the media resolution specified will be used.

model_selection_config: `Optional[ModelSelectionConfigDict]`

Optional. Config for model selection.

presence_penalty: `Optional[float]`

Optional. Positive penalties.

response_json_schema: `Optional[Any]`

- \$id - \$defs - \$ref - \$anchor - type - format - title - description - enum (for strings and numbers) - items - prefixItems - minItems - maxItems - minimum - maximum - anyOf - oneOf (interpreted the same as anyOf) - properties - additionalProperties - required The non-standard *propertyOrdering* property may also be set. Cyclic references are unrolled to a limited degree and, as such, may only be used within non-required properties. (Nullable properties are not sufficient.) If \$ref is set on a sub-schema, no other properties, except for those starting as a \$, may be set.

Skip to content

onal. Output schema of the generated response. This is an alternative to `response_schema` that accepts [JSON Schema](<https://googleapis.github.io/python-genai/genai.html#genai.types.LiveConnectConfig>)

TYPE:

//json-schema.org/). If set, `response_schema` must be omitted, but `response_mime_type` is required. While the full JSON Schema may be sent, not all features are supported. Specifically, only the following properties are supported

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_modalities: `Optional [list [Modality]]`

Optional. The modalities of the response.

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://openapi.org/specification/v3.0.3/#schema)

TYPE:

//spec.openapis.org/oas/v3.0.3#schema). If set, a compatible `response_mime_type` must also be set. Compatible mimetypes

routing_config: `Optional [GenerationConfigRoutingConfigDict]`

Optional. Routing configuration.

seed: `Optional [int]`

Optional. Seed.

***ig**: `Optional [SpeechConfigDict]`

Skip to content . The speech generation config.

stop_sequences: `Optional [list [str]]`

Optional. Stop sequences.

temperature: `Optional [float]`

Optional. Controls the randomness of predictions.

thinking_config: `Optional [GenerationConfigThinkingConfigDict]`

Optional. Config for thinking features. An error will be returned if this field is set for models that don't support thinking.

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`

Skip to content `odel`

When automated routing is specified, the routing will be determined by the pretrained routing model and customer provided model routing preference.

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:

- `model_routing_preference` (Literal['UNKNOWN', 'PRIORITYIZE_QUALITY', 'BALANCED', 'PRIORITYIZE_COST'] | None)

field `model_routing_preference`: Optional[Literal['UNKNOWN', 'PRIORITYIZE_QUALITY', 'BALANCED', 'PRIORITYIZE_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', 'PRIORITYIZE_QUALITY', 'BALANCED', 'PRIORITYIZE_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.

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_name (str | None)`

field `model_name`: Optional [str] = None (alias 'modelName')

The model name to use. Only the public LLM models are accepted. See [Supported models](<https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/inference#supported-models>).

class `genai.types.GenerationConfigRoutingConfigManualRoutingModeDict`

Bases: `TypedDict`

When manual routing is set, the specified model will be used directly.

`model_name: Optional [str]`

//cloud.google.com/vertex-ai/generative-ai/docs/model-reference/inference#supported-models).

TYPE:

The model name to use. Only the public LLM models are accepted. See [Supported models](<https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/inference#supported-models>)

pydantic model `genai.types.GenerationConfigThinkingConfig`

Bases: `BaseModel`

Config for thinking features.

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:

- `include_thoughts (bool | None)`
- `thinking_budget (int | None)`

field `include_thoughts`: `Optional[bool] = None (alias 'includeThoughts')`

Optional. Indicates whether to include thoughts in the response. If true, thoughts are returned only when available.

field `thinking_budget`: `Optional[int] = None (alias 'thinkingBudget')`

Optional. Indicates the thinking budget in tokens. This is only applied when `enable_thinking` is true.

class `genai.types.GenerationConfigThinkingConfigDict`

Bases: `TypedDict`

Config for thinking features.

`include_thoughts`: `Optional[bool]`

Optional. Indicates whether to include thoughts in the response. If true, thoughts are returned only when available.

`thinking_budget`: `Optional[int]`

Optional. Indicates the thinking budget in tokens. This is only applied when `enable_thinking` is true.

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)`

Skip to content **`options`:** `Optional[HttpOptions] = None (alias 'httpOptions')`

Used to override HTTP request options.

class genai.types.GetBatchJobConfigDictBases: `TypedDict`

Optional parameters.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

pydantic model genai.types.GetCachedContentConfigBases: `BaseModel`

Optional parameters for caches.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.GetCachedContentConfigDictBases: `TypedDict`

Optional parameters for caches.get method.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

pydantic model genai.types.GetFileConfigBases: `BaseModel`

Used to override the default configuration.

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

Skip to content `[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`

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.GetModelConfigDict`

Bases: `TypedDict`

Optional parameters for models.get method.

`http_options`: `Optional[HttpOptionsDict]`

override HTTP request options.

Skip to content

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`

Optional parameters 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`

Skip to content

Optional parameters for tunings.get method.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

pydantic model `genai.types.GoogleMaps`

Bases: `BaseModel`

Tool to support Google Maps in 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:

- `auth_config (genai.types.AuthConfig | None)`

field `auth_config`: `Optional[AuthConfig] = None (alias 'authConfig')`

Optional. Auth config for the Google Maps tool.

class `genai.types.GoogleMapsDict`

Bases: `TypedDict`

Tool to support Google Maps in Model.

auth_config: `Optional[AuthConfigDict]`

Optional. Auth config for the Google Maps tool.

pydantic model `genai.types.GoogleRpcStatus`

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>).

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:

- `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

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.

Skip to content

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

FIELDS:

- `time_range_filter` (`genai.types.Interval` | `None`)

field `time_range_filter`: `Optional[Interval] = None` (alias 'timeRangeFilter')

Optional. Filter search results to a specific time range. If customers set a start time, they must set an end time (and vice versa).

class genai.types.GoogleSearchDict

Bases: `TypedDict`

Tool to support Google Search in Model. Powered by Google.

`time_range_filter`: `Optional[IntervalDict]`

Optional. Filter search results to a specific time range. If customers set a start time, they must set an end time (and vice versa).

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.

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

► Show JSON schema

Skip to content

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`

[Skip to content](#)

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.

`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.

Skip to content

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

`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:

- `rag_chunk` (`genai.types.RagChunk | None`)
- `text` (`str | None`)
- `title` (`str | None`)
- `uri` (`str | None`)

field `rag_chunk`: `Optional[RagChunk] = None` (alias 'ragChunk')

Additional context for the RAG retrieval result. This is only populated when using the RAG retrieval tool.

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.

Skip to content `... .GroundingChunkRetrievedContextDict`

Chunk from context retrieved by the retrieval tools.

rag_chunk: `Optional[RagChunkDict]`

Additional context for the RAG retrieval result. This is only populated when using the RAG retrieval tool.

text: `Optional[str]`

Text of the attribution.

title: `Optional[str]`

Title of the attribution.

uri: `Optional[str]`

URI reference 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:

- `domain (str | None)`
- `title (str | None)`
- `uri (str | None)`

field domain: `Optional[str] = None`

Domain of the (original) URI.

field title: `Optional[str] = None`

Title of the chunk.

field uri: `Optional[str] = None`

ence of the chunk.

Skip to content

class `genai.types.GroundingChunkWebDict`

Bases: `TypedDict`

Chunk from the web.

domain: `Optional[str]`

Domain of the (original) URI.

title: `Optional[str]`

Title of the chunk.

uri: `Optional[str]`

URI reference of the chunk.

pydantic model `genai.types.GroundingMetadata`

[Skip to content](#)

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

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')`

Skip to content Web search queries for the following-up web search.

class `genai.types.GroundingMetadataDict`

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`

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

Skip to content

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. For Gemini 2.0 and before, this list must have the same size as the grounding_chunk_indices. For Gemini 2.5 and after, this list will be empty and should be ignored.

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. For Gemini 2.0 and before, this list must have the same size as the grounding_chunk_indices. For Gemini 2.5 and after, this list will be empty and should be ignored.

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 **nes.HarmBlockMethod(*values)**

base. `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'

The harm block method is unspecified.

PROBABILITY = 'PROBABILITY'

The harm block method uses the probability score.

SEVERITY = 'SEVERITY'

The harm block method uses both probability and severity scores.

class genai.types.HarmBlockThreshold(*values)

Bases: CaseInsensitiveEnum

Required. The harm block threshold.

BLOCK_LOW_AND ABOVE = 'BLOCK_LOW_AND ABOVE'

Block low threshold and above (i.e. block more).

BLOCK_MEDIUM_AND ABOVE = 'BLOCK_MEDIUM_AND ABOVE'

Block medium threshold and above.

BLOCK_NONE = 'BLOCK_NONE'

Block none.

BLOCK_ONLY_HIGH = 'BLOCK_ONLY_HIGH'

Block only high threshold (i.e. block less).

HARM_BLOCK_THRESHOLD_UNSPECIFIED = 'HARM_BLOCK_THRESHOLD_UNSPECIFIED'

Unspecified harm block threshold.

OFF = 'OFF'

Turn off the safety filter.

class genai.types.HarmCategory(*values)

Bases: CaseInsensitiveEnum

Skip to content

Required. Harm category.

HARM_CATEGORY_CIVIC_INTEGRITY = 'HARM_CATEGORY_CIVIC_INTEGRITY'

The election filter is not longer supported. The harm category is civic integrity.

TYPE:

Deprecated

HARM_CATEGORY_DANGEROUS_CONTENT = 'HARM_CATEGORY_DANGEROUS_CONTENT'

The harm category is dangerous content.

HARM_CATEGORY_HARASSMENT = 'HARM_CATEGORY_HARASSMENT'

The harm category is harassment.

HARM_CATEGORY_HATE_SPEECH = 'HARM_CATEGORY_HATE_SPEECH'

The harm category is hate speech.

HARM_CATEGORY_IMAGE_DANGEROUS_CONTENT = 'HARM_CATEGORY_IMAGE_DANGEROUS_CONTENT'

The harm category is image dangerous content.

HARM_CATEGORY_IMAGE_HARASSMENT = 'HARM_CATEGORY_IMAGE_HARASSMENT'

The harm category is image harassment.

HARM_CATEGORY_IMAGE_HATE = 'HARM_CATEGORY_IMAGE_HATE'

The harm category is image hate.

HARM_CATEGORY_IMAGE_SEXUALLY_EXPLICIT = 'HARM_CATEGORY_IMAGE_SEXUALLY_EXPLICIT'

The harm category is image sexually explicit content.

HARM_CATEGORY_SEXUALLY_EXPLICIT = 'HARM_CATEGORY_SEXUALLY_EXPLICIT'

The harm category is sexually explicit content.

HARM_CATEGORY_UNSPECIFIED = 'HARM_CATEGORY_UNSPECIFIED'

The harm category is unspecified.

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

Skip to content

Bases: CaseInsensitiveEnum

Output only. Harm probability levels in the content.

HARM_PROBABILITY_UNSPECIFIED = 'HARM_PROBABILITY_UNSPECIFIED'

Harm probability unspecified.

HIGH = 'HIGH'

High level of harm.

LOW = 'LOW'

Low level of harm.

MEDIUM = 'MEDIUM'

Medium level of harm.

NEGLIGIBLE = 'NEGLIGIBLE'

Negligible level of harm.

class genai.types.HarmSeverity(*values)

Bases: CaseInsensitiveEnum

Output only. Harm severity levels in the content.

HARM_SEVERITY_HIGH = 'HARM_SEVERITY_HIGH'

High level of harm severity.

HARM_SEVERITY_LOW = 'HARM_SEVERITY_LOW'

Low level of harm severity.

HARM_SEVERITY_MEDIUM = 'HARM_SEVERITY_MEDIUM'

Medium level of harm severity.

HARM_SEVERITY_NEGLIGIBLE = 'HARM_SEVERITY_NEGLIGIBLE'

Negligible level of harm severity.

HARM_SEVERITY_UNSPECIFIED = 'HARM_SEVERITY_UNSPECIFIED'

Harm severity unspecified.

pydantic model genai.types.HttpOptions

Model

Skip to content

.... 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)
- `async_client_args` (dict[str, Any] | None)
- `base_url` (str | None)
- `client_args` (dict[str, Any] | None)
- `extra_body` (dict[str, Any] | None)
- `headers` (dict[str, str] | None)
- `retry_options` (genai.types.HttpRetryOptions | None)
- `timeout` (int | None)

field `api_version`: Optional[str] = None (alias 'apiVersion')

Specifies the version of the API to use.

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

Args passed to the async HTTP client.

field `base_url`: Optional[str] = None (alias 'baseUrl')

The base URL for the AI platform service endpoint.

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

Args passed to the HTTP client.

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

Extra parameters to add to the request body. The structure must match the backend API's request structure. - VertexAI backend API docs: <https://cloud.google.com/vertex-ai/docs/reference/rest> - GeminiAPI backend API docs: <https://ai.google.dev/api/rest>

field `headers`: Optional[dict[str, str]] = None

Additional HTTP headers to be sent with the request.

field `retry_options`: Optional[HttpRetryOptions] = None (alias 'retryOptions')

Skip to content ry options for the request.

field `timeout`: Optional[int] = None

Timeout for the request in milliseconds.

class genai.types.HttpOptionsDict

Bases: `TypedDict`

HTTP options to be used in each of the requests.

api_version: `Optional[str]`

Specifies the version of the API to use.

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

Args passed to the async HTTP client.

base_url: `Optional[str]`

The base URL for the AI platform service endpoint.

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

Args passed to the HTTP client.

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

Extra parameters to add to the request body. The structure must match the backend API's request structure. - VertexAI backend API docs: <https://cloud.google.com/vertex-ai/docs/reference/rest> - GeminiAPI backend API docs: <https://ai.google.dev/api/rest>

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

Additional HTTP headers to be sent with the request.

retry_options: `Optional[HttpRetryOptionsDict]`

HTTP retry options for the request.

timeout: `Optional[int]`

Timeout for the request in milliseconds.

pydantic model genai.types.HttpResponse

Bases: `BaseModel`

A wrapper class for the http 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 ↗ Only positional-only to allow `self` as a field name.

▶ Show JSON schema

FIELDS:

- `body (str | None)`
- `headers (dict[str, str] | None)`

field body: `Optional[str] = None`

The raw HTTP response body, in JSON format.

field headers: `Optional[dict[str, str]] = None`

Used to retain the processed HTTP headers in the response.

class genai.types.HttpResponseDict

Bases: `TypedDict`

A wrapper class for the http response.

body: `Optional[str]`

The raw HTTP response body, in JSON format.

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

Used to retain the processed HTTP headers in the response.

pydantic model genai.types.HttpRetryOptions

Bases: `BaseModel`

HTTP retry 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

Skip to content

FIELDS:

- `attempts (int | None)`
- `exp_base (float | None)`
- `http_status_codes (list[int] | None)`
- `initial_delay (float | None)`
- `jitter (float | None)`
- `max_delay (float | None)`

field attempts: `Optional[int] = None`

Maximum number of attempts, including the original request. If 0 or 1, it means no retries.

field exp_base: `Optional[float] = None (alias 'expBase')`

Multiplier by which the delay increases after each attempt.

field http_status_codes: `Optional[list[int]] = None (alias 'httpStatusCodes')`

List of HTTP status codes that should trigger a retry. If not specified, a default set of retryable codes may be used.

field initial_delay: `Optional[float] = None (alias 'initialDelay')`

Initial delay before the first retry, in fractions of a second.

field jitter: `Optional[float] = None`

Randomness factor for the delay.

field max_delay: `Optional[float] = None (alias 'maxDelay')`

Maximum delay between retries, in fractions of a second.

class genai.types.HttpRetryOptionsDict

Skip to content

Bases: `TypedDict`

HTTP retry options to be used in each of the requests.

attempts: `Optional[int]`

Maximum number of attempts, including the original request. If 0 or 1, it means no retries.

exp_base: `Optional[float]`

Multiplier by which the delay increases after each attempt.

http_status_codes: `Optional[list[int]]`

List of HTTP status codes that should trigger a retry. If not specified, a default set of retryable codes may be used.

initial_delay: `Optional[float]`

Initial delay before the first retry, in fractions of a second.

jitter: `Optional[float]`

Randomness factor for the delay.

max_delay: `Optional[float]`

Maximum delay between retries, in fractions of a second.

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.

Skip to content

`init(context, ...)`

This function is meant to behave like a `BaseModel` method to initialise private attributes.

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.

RETURN TYPE:

None

PARAMETERS:

location – Local path where to save the image.

`show()`

Shows the image.

This method only works in a notebook environment.

RETURN TYPE:

None

`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.

`yes.ImagePromptLanguage(*values)`

Skip to content

`nSensitiveEnum`

Enum that specifies the language of the text in the prompt.

auto = 'auto'

Auto-detect the language.

en = 'en'

English

es = 'es'

Spanish

hi = 'hi'

Hindi

ja = 'ja'

Japanese

ko = 'ko'

Korean

pt = 'pt'

Portuguese

zh = 'zh'

Chinese

pydantic model genai.types.InlinedRequest

Bases: `BaseModel`

Config for inlined 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

Skip to content

FIELDS:

- config (genai.types.GenerateContentConfig | None)
 - 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)
 - model (str | None)

field config: Optional[GenerateContentConfig] = None

Configuration that contains optional model parameters.

```
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
```

Content of the request.

field model: Optional[str] = None

ID of the model to use. For a list of models, see Google models.

class genai.types.InlinedRequestDict

Bases: TypedDict

Config for inlined request.

config: Optional [GenerateContentConfigDict]

Configuration that contains optional model parameters.

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]

Content of the request.

model: Optional [str]

ID of the model to use. For a list of models, see Google models.

`pydantic model genai.types.InlinedResponse`

Bases: BaseModel

Customized responses parameter.

[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:

- `error` (`genai.types.JobError | None`)
- `response` (`genai.types.GenerateContentResponse | None`)

field `error`: `Optional[JobError] = None`

The error encountered while processing the request.

field `response`: `Optional[GenerateContentResponse] = None`

The response to the request.

`class genai.types.InlinedResponseDict`

Bases: `TypedDict`

Config for `inlined_responses` parameter.

`error`: `Optional[JobErrorDict]`

The error encountered while processing the request.

`response`: `Optional[GenerateContentResponseDict]`

The response to the request.

`pydantic model genai.types.Interval`

Bases: `BaseModel`

Represents a time interval, encoded as a start time (inclusive) and an end time (exclusive).

The start time must be less than or equal to the end time. When the start equals the end time, the interval is an empty interval. (matches no time) When both start and end are unspecified, the interval matches any 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.

Skip to content

► Show JSON schema

FIELDS:

- `end_time` (`datetime.datetime | None`)
- `start_time` (`datetime.datetime | None`)

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

The end time of the interval.

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

The start time of the interval.

class `genai.types.IntervalDict`

Bases: `TypedDict`

Represents a time interval, encoded as a start time (inclusive) and an end time (exclusive).

The start time must be less than or equal to the end time. When the start equals the end time, the interval is an empty interval. (matches no time) When both start and end are unspecified, the interval matches any time.

`end_time`: `Optional[datetime]`

The end time of the interval.

`start_time`: `Optional[datetime]`

The start time of the interval.

pydantic model `genai.types.JSONSchema`

Bases: `BaseModel`

A subset of JSON Schema according to 2020-12 JSON Schema draft.

Represents a subset of a JSON Schema object that is used by the Gemini model. The difference between this class and the `Schema` class is that this class is compatible with OpenAPI 3.1 schema objects. And the `Schema` class is used to make API call to Gemini 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](#)

Skip to content

FIELDS:

- `any_of (list[genai.types.JSONSchema] | None)`
- `default (Any | None)`
- `description (str | None)`
- `enum (list[Any] | None)`
- `format (str | None)`
- `items (genai.types.JSONSchema | 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)`
- `pattern (str | None)`
- `properties (dict[str, genai.types.JSONSchema] | None)`
- `required (list[str] | None)`
- `title (str | None)`
- `type (genai.types.JSONSchemaType | list[genai.types.JSONSchemaType] | None)`

field `any_of`: `Optional[list[JSONSchema]] = None`

An instance validates successfully against this keyword if it validates successfully against at least one schema defined by this keyword's value.

field `default`: `Optional[Any] = None`

This keyword can be used to supply a default JSON value associated with a particular schema.

field `description`: `Optional[str] = None`

An explanation about the purpose of the instance described by the schema.

field `enum`: `Optional[list[Any]] = None`

Validation succeeds if the instance is equal to one of the elements in this keyword's array value.

Skip to content

t: `Optional[str] = None`

Define semantic information about a string instance.

field items: Optional[JSONSchema] = None

Validation succeeds if each element of the instance not covered by prefixItems validates against this schema.

field max_items: Optional[int] = None

An array instance is valid if its size is less than, or equal to, the value of this keyword.

field max_length: Optional[int] = None

A string instance is valid against this keyword if its length is less than, or equal to, the value of this keyword.

field max_properties: Optional[int] = None

An object instance is valid if its number of properties is less than, or equal to, the value of this keyword.

field maximum: Optional[float] = None

Validation succeeds if the numeric instance is less than or equal to the given number.

field min_items: Optional[int] = None

An array instance is valid if its size is greater than, or equal to, the value of this keyword.

field min_length: Optional[int] = None

A string instance is valid against this keyword if its length is greater than, or equal to, the value of this keyword.

field min_properties: Optional[int] = None

An object instance is valid if its number of properties is greater than, or equal to, the value of this keyword.

field minimum: Optional[float] = None

Validation succeeds if the numeric instance is greater than or equal to the given number.

field pattern: Optional[str] = None

A string instance is considered valid if the regular expression matches the instance successfully.

field properties: Optional[dict[str, JSONSchema]] = None

Validation succeeds if, for each name that appears in both the instance and as a name within this keyword's value, the child instance for that name successfully validates against

Skip to content corresponding schema.

field required: Optional[list[str]] = None

An object instance is valid against this keyword if every item in the array is the name of a property in the instance.

field title: Optional [str] = None

A preferably short description about the purpose of the instance described by the schema.

field type: Union [JSONSchemaType , list [JSONSchemaType] , None] = None

Validation succeeds if the type of the instance matches the type represented by the given type, or matches at least one of the given types.

class genai.types.JSONSchemaType(*values)

Bases: `Enum`

The type of the data supported by JSON Schema.

The values of the enums are lower case strings, while the values of the enums for the Type class are upper case strings.

ARRAY = 'array'

BOOLEAN = 'boolean'

INTEGER = 'integer'

NULL = 'null'

NUMBER = 'number'

OBJECT = 'object'

STRING = 'string'

pydantic model 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

Skip to content

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'

The job has been cancelled.

JOB_STATE_CANCELLING = 'JOB_STATE_CANCELLING'

The job is being cancelled. From this state the job may only go to either *JOB_STATE_SUCCEEDED*, *JOB_STATE_FAILED* or *JOB_STATE_CANCELLED*.

JOB_STATE_EXPIRED = 'JOB_STATE_EXPIRED'

The job has expired.

JOB_STATE_FAILED = 'JOB_STATE_FAILED'

The job failed.

JOB_STATE_PARTIALLY_SUCCEEDED = 'JOB_STATE_PARTIALLY_SUCCEEDED'

The job is partially succeeded, some results may be missing due to errors.

JOB_STATE_PAUSED = 'JOB_STATE_PAUSED'

The job has been stopped, and can be resumed.

JOB_STATE_PENDING = 'JOB_STATE_PENDING'

The service is preparing to run the job.

JOB_STATE_QUEUED = 'JOB_STATE_QUEUED'

The job has been just created or resumed and processing has not yet begun.

JOB_STATE_RUNNING = 'JOB_STATE_RUNNING'

The job is in progress.

JOB_STATE_SUCCEEDED = 'JOB_STATE_SUCCEEDED'

The job completed successfully.

JOB_STATE_UNSPECIFIED = 'JOB_STATE_UNSPECIFIED'

The job state is unspecified.

JOB_STATE_UPDATING = 'JOB_STATE_UPDATING'

The job is being updated. Only jobs in the *JOB_STATE_RUNNING* state can be updated.

Skip to content Later, the job goes back to the *JOB_STATE_RUNNING* state.

class genai.types.Language(*values)

Bases: CaseInsensitiveEnum

Required. Programming language of the code.

LANGUAGE_UNSPECIFIED = 'LANGUAGE_UNSPECIFIED'

Unspecified language. This value should not be used.

PYTHON = 'PYTHON'

Python >= 3.10, with numpy and simpy available.

pydantic model genai.types.LatLng

Bases: BaseModel

An object that represents a latitude/longitude pair.

This is expressed as a pair of doubles to represent degrees latitude and degrees longitude.

Unless specified otherwise, this object must conform to the WGS84

standard. Values must be within normalized ranges.

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:

- `latitude (float | None)`
- `longitude (float | None)`

field latitude: Optional[float] = None

The latitude in degrees. It must be in the range [-90.0, +90.0].

field longitude: Optional[float] = None

The longitude in degrees. It must be in the range [-180.0, +180.0]

class genai.types.LatLngDict

Bases: TypedDict

t represents a latitude/longitude pair.

Skip to content

This is expressed as a pair of doubles to represent degrees latitude and degrees longitude. Unless specified otherwise, this object must conform to the WGS84 standard. Values must be within normalized ranges.

latitude: Optional [float]

The latitude in degrees. It must be in the range [-90.0, +90.0].

longitude: Optional [float]

The longitude in degrees. It must be in the range [-180.0, +180.0]

pydantic model genai.types.ListBatchJobsConfig

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

Skip to content

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.

`self` is explicitly 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`

Skip to content

↳ ... ↳ `batches.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]`

Used to override HTTP request options.

`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

Skip to content

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`

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`')

Skip to content **`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

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.

`token: Optional[str]`

Skip to content

.o 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')`

`field page_token: Optional[str] = None (alias 'pageToken')`

`field query_base: Optional[bool] = None (alias 'queryBase')`

Set true to list base models, false to list tuned models.

`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.

Skip to content `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
```

Skip to content

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.

`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.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`

Skip to content 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 obtain that page.

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.

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`

nt by the client in the API call.

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:

- `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`')

Response 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.

Skip to content

`tool_response`: `Optional[LiveClientToolResponseDict]`

Response to a `ToolCallMessage` received from the server.

pydantic model genai.types.LiveClientRealtimeInput

[Skip to content](#)

Bases: `BaseModel`

User input that is sent in real time.

This is different from `LiveClientContent` in a few ways:

- Can be sent continuously without interruption to model generation.
- If there is a need to mix data interleaved across the `LiveClientContent` and the `LiveClientRealtimeInput`, 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

FIELDS:

- `activity_end` (`genai.types.ActivityEnd | None`)
- `activity_start` (`genai.types.ActivityStart | None`)
- `audio` (`genai.types.Blob | None`)
- `audio_stream_end` (`bool | None`)
- `media_chunks` (`list[genai.types.Blob] | None`)
- `text` (`str | None`)
- `video` (`genai.types.Blob | None`)

```
field activity_end: Optional[ActivityEnd] = None (alias 'activityEnd')
```

Marks the end of user activity.

```
Skip to content activity_start: Optional[ActivityStart] = None (alias 'activityStart')
```

Marks the start of user activity.

```
field audio: Optional[Blob] = None
```

The realtime audio input stream.

```
field audio_stream_end: Optional[bool] = None (alias 'audioStreamEnd')
```

Indicates that the audio stream has ended, e.g. because the microphone was turned off.

This should only be sent when automatic activity detection is enabled (which is the default).

The client can reopen the stream by sending an audio message.

```
field media_chunks: Optional[list[Blob]] = None (alias 'mediaChunks')
```

Inlined bytes data for media input.

```
field text: Optional[str] = None
```

The realtime text input stream.

```
field video: Optional[Blob] = None
```

The realtime video input stream.

```
class genai.types.LiveClientRealtimeInputDict
```

[Skip to content](#)

Bases: `TypedDict`

User input that is sent in real time.

This is different from `LiveClientContent` in a few ways:

- Can be sent continuously without interruption to model generation.
- If there is a need to mix data interleaved across the `LiveClientContent` and the `LiveClientRealtimeInput`, 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).

activity_end: `Optional[ActivityEndDict]`

Marks the end of user activity.

activity_start: `Optional[ActivityStartDict]`

Marks the start of user activity.

audio: `Optional[BlobDict]`

The realtime audio input stream.

audio_stream_end: `Optional[bool]`

Indicates that the audio stream has ended, e.g. because the microphone was turned off.

This should only be sent when automatic activity detection is enabled (which is the default).

The client can reopen the stream by sending an audio message.

media_chunks: `Optional[list[BlobDict]]`

Inlined bytes data for media input.

text: `Optional[str]`

Skip to contentime text input stream.

video: `Optional[BlobDict]`

The realtime video input stream.

pydantic model genai.types.LiveClientSetup

[Skip to content](#)

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

FIELDS:

- `context_window_compression` (`genai.types.ContextWindowCompressionConfig | None`)
- `generation_config` (`genai.types.GenerationConfig | None`)
- `input_audio_transcription` (`genai.types.AudioTranscriptionConfig | None`)
- `model` (`str | None`)
- `output_audio_transcription` (`genai.types.AudioTranscriptionConfig | None`)
- `proactivity` (`genai.types.ProactivityConfig | None`)
- `session_resumption` (`genai.types.SessionResumptionConfig | 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 | Callable[[], Any] | mcp.types.Tool | mcp.client.session.ClientSession] | None`)

field `context_window_compression`: `Optional[ContextWindowCompressionConfig] = None`
`(alias 'contextWindowCompression')`

Configures context window compression mechanism.

If included, server will compress context window to fit into given length.

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

The generation configuration for the session. Note: only a subset of fields are supported.

field `input_audio_transcription`: `Optional[AudioTranscriptionConfig] = None` `(alias 'inputAudioTranscription')`

The transcription of the input aligns with the input audio language.

`Skip to content : Optional[str] = None`

The fully qualified name of the publisher model or tuned model endpoint to use.

```
field output_audio_transcription: Optional[AudioTranscriptionConfig] = None (alias  
'outputAudioTranscription')
```

The transcription of the output aligns with the language code specified for the output audio.

```
field proactivity: Optional[ProactivityConfig] = None
```

Configures the proactivity of the model. This allows the model to respond proactively to the input and to ignore irrelevant input.

```
field session_resumption: Optional[SessionResumptionConfig] = None (alias  
'sessionResumption')
```

Configures session resumption mechanism.

If included server will send SessionResumptionUpdate messages.

```
field system_instruction: Union[Content, list[Union[File, Part, Image, str]],  
File, Part, Image, str, None] = 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[..., Any], Tool, ClientSession]]] =  
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
```

Skip to content

Bases: `TypedDict`

Message contains configuration that will apply for the duration of the streaming session.

context_window_compression: `Optional[ContextWindowCompressionConfigDict]`

Configures context window compression mechanism.

If included, server will compress context window to fit into given length.

generation_config: `Optional[GenerationConfigDict]`

The generation configuration for the session. Note: only a subset of fields are supported.

input_audio_transcription: `Optional[AudioTranscriptionConfigDict]`

The transcription of the input aligns with the input audio language.

model: `Optional[str]`

The fully qualified name of the publisher model or tuned model endpoint to use.

output_audio_transcription: `Optional[AudioTranscriptionConfigDict]`

The transcription of the output aligns with the language code specified for the output audio.

proactivity: `Optional[ProactivityConfigDict]`

Configures the proactivity of the model. This allows the model to respond proactively to the input and to ignore irrelevant input.

session_resumption: `Optional[SessionResumptionConfigDict]`

Configures session resumption mechanism.

If included server will send SessionResumptionUpdate messages.

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

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[..., Any], Tool, ClientSession]]]`

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.

Skip to content

[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`

Skip to content  for the API connection.

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:

- `context_window_compression` (`genai.types.ContextWindowCompressionConfig | None`)
- `enable_affective_dialog` (`bool | None`)
- `generation_config` (`genai.types.GenerationConfig | None`)
- `http_options` (`genai.types.HttpOptions | None`)
- `input_audio_transcription` (`genai.types.AudioTranscriptionConfig | None`)
- `max_output_tokens` (`int | None`)
- `media_resolution` (`genai.types.MediaResolution | None`)
- `output_audio_transcription` (`genai.types.AudioTranscriptionConfig | None`)
- `proactivity` (`genai.types.ProactivityConfig | None`)
- `realtime_input_config` (`genai.types.RealtimeInputConfig | None`)
- `response_modalities` (`list[genai.types.Modality] | None`)
- `seed` (`int | None`)
- `session_resumption` (`genai.types.SessionResumptionConfig | None`)
- `speech_config` (`genai.types.SpeechConfig | 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`)
- `tools` (`list[genai.types.Tool | Callable[[], Any] | mcp.types.Tool | mcp.client.session.ClientSession] | None`)
- `top_k` (`float | None`)
- `top_p` (`float | None`)

**field `context_window_compression`: Optional[`ContextWindowCompressionConfig`] = None
(alias '`contextWindowCompression`')**

Configures context window compression mechanism.

If included, server will compress context window to fit into given length.

Skip to content **`e_affective_dialog`: Optional[`bool`] = None (alias '`eAffectiveDialog`')**

If enabled, the model will detect emotions and adapt its responses accordingly.

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

The generation configuration for the session.

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

Used to override HTTP request options.

```
field input_audio_transcription: Optional[AudioTranscriptionConfig] = None (alias  
'inputAudioTranscription')
```

The transcription of the input aligns with the input audio language.

```
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 output_audio_transcription: Optional[AudioTranscriptionConfig] = None (alias  
'outputAudioTranscription')
```

The transcription of the output aligns with the language code specified for the output audio.

```
field proactivity: Optional[ProactivityConfig] = None
```

Configures the proactivity of the model. This allows the model to respond proactively to the input and to ignore irrelevant input.

```
field realtime_input_config: Optional[RealtimeInputConfig] = None (alias  
'realtimeInputConfig')
```

Configures the realtime input behavior in BidiGenerateContent.

```
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 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 session_resumption: Optional[SessionResumptionConfig] = None (alias
```

Skip to content `'onResumption'`)

Configures session resumption mechanism.

If included the server will send SessionResumptionUpdate messages.

field speech_config: `Optional[SpeechConfig] = None (alias 'speechConfig')`

The speech generation configuration.

field system_instruction: `Union[Content, list[Union[File, Part, Image, str]], File, Part, Image, str, None] = 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 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 tools: `Optional[list[Union[Tool, Callable[..., Any], Tool, ClientSession]]] = 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.

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 random responses.

class genai.types.LiveConnectConfigDict

Skip to content

Bases: `TypedDict`

Session config for the API connection.

context_window_compression: `Optional[ContextWindowCompressionConfigDict]`

Configures context window compression mechanism.

If included, server will compress context window to fit into given length.

enable_affective_dialog: `Optional[bool]`

If enabled, the model will detect emotions and adapt its responses accordingly.

generation_config: `Optional[GenerationConfigDict]`

The generation configuration for the session.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

input_audio_transcription: `Optional[AudioTranscriptionConfigDict]`

The transcription of the input aligns with the input audio language.

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.

output_audio_transcription: `Optional[AudioTranscriptionConfigDict]`

The transcription of the output aligns with the language code specified for the output audio.

proactivity: `Optional[ProactivityConfigDict]`

Configures the proactivity of the model. This allows the model to respond proactively to the input and to ignore irrelevant input.

realtime_input_config: `Optional[RealtimeInputConfigDict]`

Configures the realtime input behavior in BidiGenerateContent.

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.

Skip to content

-----: `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.

session_resumption: `Optional[SessionResumptionConfigDict]`

Configures session resumption mechanism.

If included the server will send SessionResumptionUpdate messages.

speech_config: `Optional[SpeechConfigDict]`

The speech generation configuration.

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

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.

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.

tools: `Optional[list[Union[ToolDict, Callable[..., Any]], Tool, ClientSession]]`

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.

top_k: `Optional[float]`

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.

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.LiveConnectConstraints

Bases: `BaseModel`

Skip to content eConnectConstraints for Auth Token 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:

- `config` (`genai.types.LiveConnectConfig | None`)
- `model` (`str | None`)

field config: `Optional[LiveConnectConfig] = None`

Configuration specific to Live API connections created using this token.

field model: `Optional[str] = None`

ID of the model to configure in the ephemeral token for Live API. For a list of models, see *Gemini models* <<https://ai.google.dev/gemini-api/docs/models>>.

class genai.types.LiveConnectConstraintsDict

Bases: `TypedDict`

Config for LiveConnectConstraints for Auth Token creation.

config: `Optional[LiveConnectConfigDict]`

Configuration specific to Live API connections created using this token.

model: `Optional[str]`

ID of the model to configure in the ephemeral token for Live API. For a list of models, see *Gemini models* <<https://ai.google.dev/gemini-api/docs/models>>.

pydantic model genai.types.LiveConnectParameters

Bases: `BaseModel`

Parameters for connecting to the live 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.

↗ schema

Skip to content

FIELDS:

- `config (genai.types.LiveConnectConfig | None)`
- `model (str | None)`

field config: `Optional[LiveConnectConfig] = None`

Optional configuration parameters for the request.

field model: `Optional[str] = None`

ID of the model to use. For a list of models, see Google models.

class genai.types.LiveConnectParametersDict

Bases: `TypedDict`

Parameters for connecting to the live API.

config: `Optional[LiveConnectConfigDict]`

Optional configuration parameters for the request.

model: `Optional[str]`

ID of the model to use. For a list of models, see Google models.

pydantic model genai.types.LiveMusicClientContent

Bases: `BaseModel`

User input to start or steer the music.

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:

- `weighted_prompts (list[genai.types.WeightedPrompt] | None)`

field weighted_prompts: `Optional[list[WeightedPrompt]] = None (alias 'weightedPrompts')`

Weighted prompts as the model input.

Skip to content **genai.types.LiveMusicClientContentDict**

Bases: `TypedDict`

User input to start or steer the music.

weighted_prompts: `Optional[list[WeightedPromptDict]]`

Weighted prompts as the model input.

pydantic model genai.types.LiveMusicClientMessage

Bases: `BaseModel`

Messages sent by the client in the `LiveMusicClientMessage` 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.LiveMusicClientContent | None`)
- `music_generation_config` (`genai.types.LiveMusicGenerationConfig | None`)
- `playback_control` (`genai.types.LiveMusicPlaybackControl | None`)
- `setup` (`genai.types.LiveMusicClientSetup | None`)

field client_content: `Optional[LiveMusicClientContent] = None (alias 'clientContent')`

User input to influence music generation.

field music_generation_config: `Optional[LiveMusicGenerationConfig] = None (alias 'musicGenerationConfig')`

Configuration for music generation.

field playback_control: `Optional[LiveMusicPlaybackControl] = None (alias 'playbackControl')`

Playback control signal for the music generation.

field setup: `Optional[LiveMusicClientSetup] = None`

Message to be sent in the first (and only in the first) `LiveMusicClientMessage`. Clients should wait for a `LiveMusicSetupComplete` message before sending any additional messages.

Skip to content

`yes.LiveMusicClientMessageDict`

Bases: `TypedDict`

Messages sent by the client in the `LiveMusicClientMessage` call.

client_content: `Optional[LiveMusicClientContentDict]`

User input to influence music generation.

music_generation_config: `Optional[LiveMusicGenerationConfigDict]`

Configuration for music generation.

playback_control: `Optional[LiveMusicPlaybackControl]`

Playback control signal for the music generation.

setup: `Optional[LiveMusicClientSetupDict]`

Message to be sent in the first (and only in the first) `LiveMusicClientMessage`. Clients should wait for a `LiveMusicSetupComplete` message before sending any additional messages.

pydantic model genai.types.LiveMusicClientSetup

Bases: `BaseModel`

Message to be sent by the system when connecting 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

FIELDS:

- `model (str | None)`

field model: `Optional[str] = None`

The model's resource name. Format: `models/{model}`.

class genai.types.LiveMusicClientSetupDict

Bases: `TypedDict`

Message to be sent by the system when connecting to the API.

model: `Optional[str]`

Skip to content `{model}`.

TYPE:

The model's resource name. Format

pydantic model genai.types.LiveMusicConnectParameters

Bases: `BaseModel`

Parameters for connecting to the live 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:

- `model (str | None)`

field `model`: `Optional[str] = None`

The model's resource name.

class genai.types.LiveMusicConnectParametersDict

Bases: `TypedDict`

Parameters for connecting to the live API.

model: `Optional[str]`

The model's resource name.

pydantic model genai.types.LiveMusicFilteredPrompt

Bases: `BaseModel`

A prompt that was filtered with the reason.

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:

- `filtered_reason (str | None)`
- `text (str | None)`

field `filtered_reason`: `Optional[str] = None (alias 'filteredReason')`

The reason the prompt was filtered.

field `text`: `Optional[str] = None`

The text prompt that was filtered.

class `genai.types.LiveMusicFilteredPromptDict`

Bases: `TypedDict`

A prompt that was filtered with the reason.

`filtered_reason`: `Optional[str]`

The reason the prompt was filtered.

`text`: `Optional[str]`

The text prompt that was filtered.

pydantic model `genai.types.LiveMusicGenerationConfig`

Bases: `BaseModel`

Configuration for music 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

Skip to content

FIELDS:

- `bpm (int | None)`
- `brightness (float | None)`
- `density (float | None)`
- `guidance (float | None)`
- `mute_bass (bool | None)`
- `mute_drums (bool | None)`
- `only_bass_and_drums (bool | None)`
- `scale (genai.types.Scale | None)`
- `seed (int | None)`
- `temperature (float | None)`
- `top_k (int | None)`

field `bpm`: `Optional[int] = None`

Beats per minute. Range is [60, 200].

field `brightness`: `Optional[float] = None`

Brightness of the music. Range is [0.0, 1.0].

field `density`: `Optional[float] = None`

Density of sounds. Range is [0.0, 1.0].

field `guidance`: `Optional[float] = None`

Controls how closely the model follows prompts. Higher guidance follows more closely, but will make transitions more abrupt. Range is [0.0, 6.0].

field `mute_bass`: `Optional[bool] = None (alias 'muteBass')`

Whether the audio output should contain bass.

field `mute_drums`: `Optional[bool] = None (alias 'muteDrums')`

Whether the audio output should contain drums.

field `only_bass_and_drums`: `Optional[bool] = None (alias 'onlyBassAndDrums')`

Whether the audio output should contain only bass and drums.

field `scale`: `Optional[Scale] = None`

Scale of the generated music.

Skip to content

`Optional[int] = None`

Seeds audio generation. If not set, the request uses a randomly generated seed.

```
field temperature: Optional[ float ] = None
```

Controls the variance in audio generation. Higher values produce higher variance. Range is [0.0, 3.0].

```
field top_k: Optional[ int ] = None (alias 'topK')
```

Controls how the model selects tokens for output. Samples the topK tokens with the highest probabilities. Range is [1, 1000].

```
class genai.types.LiveMusicGenerationConfigDict
```

[Skip to content](#)

Bases: `TypedDict`

Configuration for music generation.

bpm: `Optional[int]`

Beats per minute. Range is [60, 200].

brightness: `Optional[float]`

Brightness of the music. Range is [0.0, 1.0].

density: `Optional[float]`

Density of sounds. Range is [0.0, 1.0].

guidance: `Optional[float]`

Controls how closely the model follows prompts. Higher guidance follows more closely, but will make transitions more abrupt. Range is [0.0, 6.0].

mute_bass: `Optional[bool]`

Whether the audio output should contain bass.

mute_drums: `Optional[bool]`

Whether the audio output should contain drums.

only_bass_and_drums: `Optional[bool]`

Whether the audio output should contain only bass and drums.

scale: `Optional[Scale]`

Scale of the generated music.

seed: `Optional[int]`

Seeds audio generation. If not set, the request uses a randomly generated seed.

temperature: `Optional[float]`

Controls the variance in audio generation. Higher values produce higher variance. Range is [0.0, 3.0].

top_k: `Optional[int]`

Controls how the model selects tokens for output. Samples the topK tokens with the highest probabilities. Range is [1, 1000].

Skip to content `genai.LiveMusicPlaybackControl(*values)`

`inSensitiveEnum`

The playback control signal to apply to the music generation.

PAUSE = 'PAUSE'

Hold the music generation. Use PLAY to resume from the current position.

PLAY = 'PLAY'

Start generating the music.

PLAYBACK_CONTROL_UNSPECIFIED = 'PLAYBACK_CONTROL_UNSPECIFIED'

This value is unused.

RESET_CONTEXT = 'RESET_CONTEXT'

Reset the context of the music generation without stopping it. Retains the current prompts and config.

STOP = 'STOP'

Stop the music generation and reset the context (prompts retained). Use PLAY to restart the music generation.

pydantic model `genai.types.LiveMusicServerContent`

Bases: `BaseModel`

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:

- `audio_chunks` (`list[genai.types.AudioChunk] | None`)

field `audio_chunks`: `Optional[list[AudioChunk]] = None (alias 'audioChunks')`

The audio chunks that the model has generated.

Skip to content **yes**.`LiveMusicServerContentDict`

`... Dict`

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.

audio_chunks: `Optional [list [AudioChunkDict]]`

The audio chunks that the model has generated.

pydantic model genai.types.LiveMusicServerMessage

Bases: `BaseModel`

Response message for the `LiveMusicClientMessage` 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:

- `filtered_prompt` (`genai.types.LiveMusicFilteredPrompt | None`)
- `server_content` (`genai.types.LiveMusicServerContent | None`)
- `setup_complete` (`genai.types.LiveMusicServerSetupComplete | None`)

field `filtered_prompt`: `Optional [LiveMusicFilteredPrompt] = None (alias 'filteredPrompt')`

A prompt that was filtered with the reason.

field `server_content`: `Optional [LiveMusicServerContent] = None (alias 'serverContent')`

Content generated by the model in response to client messages.

field `setup_complete`: `Optional [LiveMusicServerSetupComplete] = None (alias 'setupComplete')`

Message sent in response to a `LiveMusicClientSetup` message from the client. Clients should wait for this message before sending any additional messages.

class `genai.types.LiveMusicServerMessageDict`

Skip to content

Bases: `TypedDict`

Response message for the `LiveMusicClientMessage` call.

filtered_prompt: `Optional[LiveMusicFilteredPromptDict]`

A prompt that was filtered with the reason.

server_content: `Optional[LiveMusicServerContentDict]`

Content generated by the model in response to client messages.

setup_complete: `Optional[LiveMusicServerSetupCompleteDict]`

Message sent in response to a `LiveMusicClientSetup` message from the client. Clients should wait for this message before sending any additional messages.

pydantic model `genai.types.LiveMusicServerSetupComplete`

Bases: `BaseModel`

Sent in response to a `LiveMusicClientSetup` 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.LiveMusicServerSetupCompleteDict`

Bases: `TypedDict`

Sent in response to a `LiveMusicClientSetup` message from the client.

pydantic model `genai.types.LiveMusicSetConfigParameters`

Bases: `BaseModel`

Parameters for setting config for the live music 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.

Skip to content

▶ Show JSON schema

FIELDS:

- `music_generation_config` (`genai.types.LiveMusicGenerationConfig` | `None`)
- field `music_generation_config`:** `Optional[LiveMusicGenerationConfig] = None` (alias '`musicGenerationConfig`')
- Configuration for music generation.

class `genai.types.LiveMusicSetConfigParametersDict`

Bases: `TypedDict`

Parameters for setting config for the live music API.

`music_generation_config`: `Optional[LiveMusicGenerationConfigDict]`

Configuration for music generation.

pydantic model `genai.types.LiveMusicSetWeightedPromptsParameters`

Bases: `BaseModel`

Parameters for setting weighted prompts for the live music 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:

- `weighted_prompts` (`list[genai.types.WeightedPrompt]` | `None`)
- field `weighted_prompts`:** `Optional[list[WeightedPrompt]] = None` (alias '`weightedPrompts`')
- A map of text prompts to weights to use for the generation request.

class `genai.types.LiveMusicSetWeightedPromptsParametersDict`

Bases: `TypedDict`

Parameters for setting weighted prompts for the live music API.

`weighted_prompts`: `Optional[list[WeightedPromptDict]]`

‘text prompts to weights to use for the generation request.

Skip to content

pydantic model `genai.types.LiveMusicSourceMetadata`

Bases: `BaseModel`

Prompts and config used for generating this audio 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:

- `client_content` (genai.types.LiveMusicClientContent | None)
- `music_generation_config` (genai.types.LiveMusicGenerationConfig | None)

field `client_content`: Optional[LiveMusicClientContent] = None (alias 'clientContent')

Weighted prompts for generating this audio chunk.

field `music_generation_config`: Optional[LiveMusicGenerationConfig] = None (alias 'musicGenerationConfig')

Music generation config for generating this audio chunk.

class genai.types.LiveMusicSourceMetadataDict

Bases: TypedDict

Prompts and config used for generating this audio chunk.

client_content: Optional[LiveMusicClientContentDict]

Weighted prompts for generating this audio chunk.

music_generation_config: Optional[LiveMusicGenerationConfigDict]

Music generation config for generating this audio chunk.

pydantic model genai.types.LiveSendRealtimeInputParameters

Bases: BaseModel

Parameters for sending realtime input to the live 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.

Skip to content

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

► Show JSON schema

FIELDS:

- `activity_end` (`genai.types.ActivityEnd | None`)
- `activity_start` (`genai.types.ActivityStart | None`)
- `audio` (`genai.types.Blob | None`)
- `audio_stream_end` (`bool | None`)
- `media` (`genai.types.Blob | PIL.Image.Image | None`)
- `text` (`str | None`)
- `video` (`genai.types.Blob | PIL.Image.Image | None`)

field `activity_end`: `Optional[ActivityEnd] = None (alias 'activityEnd')`

Marks the end of user activity.

field `activity_start`: `Optional[ActivityStart] = None (alias 'activityStart')`

Marks the start of user activity.

field `audio`: `Optional[Blob] = None`

The realtime audio input stream.

field `audio_stream_end`: `Optional[bool] = None (alias 'audioStreamEnd')`

Indicates that the audio stream has ended, e.g. because the microphone was turned off.

This should only be sent when automatic activity detection is enabled (which is the default).

The client can reopen the stream by sending an audio message.

field `media`: `Union[Blob, Image, None] = None`

Realtime input to send to the session.

field `text`: `Optional[str] = None`

The realtime text input stream.

field `video`: `Union[Blob, Image, None] = None`

The realtime video input stream.

class `genai.types.LiveSendRealtimeInputParametersDict`

Skip to content

Bases: `TypedDict`

Parameters for sending realtime input to the live API.

activity_end: `Optional[ActivityEndDict]`

Marks the end of user activity.

activity_start: `Optional[ActivityStartDict]`

Marks the start of user activity.

audio: `Optional[BlobDict]`

The realtime audio input stream.

audio_stream_end: `Optional[bool]`

Indicates that the audio stream has ended, e.g. because the microphone was turned off.

This should only be sent when automatic activity detection is enabled (which is the default).

The client can reopen the stream by sending an audio message.

media: `Union[Blob, Image, BlobDict, None]`

Realtime input to send to the session.

text: `Optional[str]`

The realtime text input stream.

video: `Union[Blob, Image, BlobDict, None]`

The realtime video input stream.

pydantic model `genai.types.LiveServerContent`

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.

Skip to content ↗ Only positional-only to allow `self` as a field name.

► Show JSON schema

FIELDS:

- `generation_complete (bool | None)`
- `grounding_metadata (genai.types.GroundingMetadata | None)`
- `input_transcription (genai.types.Transcription | None)`
- `interrupted (bool | None)`
- `model_turn (genai.types.Content | None)`
- `output_transcription (genai.types.Transcription | None)`
- `turn_complete (bool | None)`
- `url_context_metadata (genai.types.UrlContextMetadata | None)`

field `generation_complete`: `Optional[bool] = None (alias 'generationComplete')`

If true, indicates that the model is done generating. When model is interrupted while generating there will be no generation_complete message in interrupted turn, it will go through interrupted > turn_complete. When model assumes realtime playback there will be delay between generation_complete and turn_complete that is caused by model waiting for playback to finish. If true, indicates that the model has finished generating all content. This is a signal to the client that it can stop sending messages.

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

Metadata returned to client when grounding is enabled.

field `input_transcription`: `Optional[Transcription] = None (alias 'inputTranscription')`

Input transcription. The transcription is independent to the model turn which means it doesn't imply any ordering between transcription and model turn.

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 `output_transcription`: `Optional[Transcription] = None (alias 'outputTranscription')`

Transcription. The transcription is independent to the model turn which means it skip to content imply any ordering between transcription and model turn.

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.

```
field url_context_metadata: Optional[UrlContextMetadata] = None (alias  
    'urlContextMetadata')
```

Metadata related to url context retrieval tool.

```
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.

generation_complete: `Optional[bool]`

If true, indicates that the model is done generating. When model is interrupted while generating there will be no generation_complete message in interrupted turn, it will go through interrupted > turn_complete. When model assumes realtime playback there will be delay between generation_complete and turn_complete that is caused by model waiting for playback to finish. If true, indicates that the model has finished generating all content. This is a signal to the client that it can stop sending messages.

grounding_metadata: `Optional[GroundingMetadataDict]`

Metadata returned to client when grounding is enabled.

input_transcription: `Optional[TranscriptionDict]`

Input transcription. The transcription is independent to the model turn which means it doesn't imply any ordering between transcription and model turn.

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.

output_transcription: `Optional[TranscriptionDict]`

Output transcription. The transcription is independent to the model turn which means it doesn't imply any ordering between transcription and model turn.

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.

url_context_metadata: `Optional[UrlContextMetadataDict]`

Skip to content ↗ related to url context retrieval tool.

pydantic model `genai.types.LiveServerGoAway`

Bases: `BaseModel`

Server will not be able to service client soon.

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:

- `time_left (str | None)`

field `time_left`: `Optional[str] = None (alias 'timeLeft')`

The remaining time before the connection will be terminated as ABORTED. The minimal time returned here is specified differently together with the rate limits for a given model.

`class genai.types.LiveServerGoAwayDict`

Bases: `TypedDict`

Server will not be able to service client soon.

time_left: `Optional[str]`

The remaining time before the connection will be terminated as ABORTED. The minimal time returned here is specified differently together with the rate limits for a given model.

`pydantic model genai.types.LiveServerMessage`

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

Skip to content

FIELDS:

- `go_away` (`genai.types.LiveServerGoAway` | `None`)
- `server_content` (`genai.types.LiveServerContent` | `None`)
- `session_resumption_update` (`genai.types.LiveServerSessionResumptionUpdate` | `None`)
- `setup_complete` (`genai.types.LiveServerSetupComplete` | `None`)
- `tool_call` (`genai.types.LiveServerToolCall` | `None`)
- `tool_call_cancellation` (`genai.types.LiveServerToolCallCancellation` | `None`)
- `usage_metadata` (`genai.types.UsageMetadata` | `None`)

field `go_away`: `Optional[LiveServerGoAway] = None (alias 'goAway')`

Server will disconnect soon.

field `server_content`: `Optional[LiveServerContent] = None (alias 'serverContent')`

Content generated by the model in response to client messages.

field `session_resumption_update`: `Optional[LiveServerSessionResumptionUpdate] = None (alias 'sessionResumptionUpdate')`

Update of the session resumption state.

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.

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

Usage metadata about model response(s).

property `data`: `bytes | None`

Returns the concatenation of all inline data parts in the response.

`txt: str | None`

Skip to content `txt`: the concatenation of all text parts in the response.

class `genai.types.LiveServerMessageDict`

Bases: `TypedDict`

Response message for API call.

go_away: `Optional[LiveServerGoAwayDict]`

Server will disconnect soon.

server_content: `Optional[LiveServerContentDict]`

Content generated by the model in response to client messages.

session_resumption_update: `Optional[LiveServerSessionResumptionUpdateDict]`

Update of the session resumption state.

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.

usage_metadata: `Optional[UsageMetadataDict]`

Usage metadata about model response(s).

pydantic model `genai.types.LiveServerSessionResumptionUpdate`

Bases: `BaseModel`

Update of the session resumption state.

Only sent if *session_resumption* was set in the connection 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

Skip to content

FIELDS:

- `last_consumed_client_message_index (int | None)`
- `new_handle (str | None)`
- `resumable (bool | None)`

field `last_consumed_client_message_index`: `Optional[int] = None (alias 'lastConsumedClientMessageIndex')`

Index of last message sent by client that is included in state represented by this `SessionResumptionToken`. Only sent when `SessionResumptionConfig.transparent` is set.

Presence of this index allows users to transparently reconnect and avoid issue of losing some part of realtime audio input/video. If client wishes to temporarily disconnect (for example as result of receiving `GoAway`) they can do it without losing state by buffering messages sent since last `SessionResumptionTokenUpdate`. This field will enable them to limit buffering (avoid keeping all requests in RAM).

Note: This should not be used for when resuming a session at some time later – in those cases partial audio and video frames are likely not needed.

field `new_handle`: `Optional[str] = None (alias 'newHandle')`

New handle that represents state that can be resumed. Empty if `resumable`=false.

field `resumable`: `Optional[bool] = None`

True if session can be resumed at this point. It might be not possible to resume session at some points. In that case we send update empty `new_handle` and `resumable=false`.

Example of such case could be model executing function calls or just generating.

Resuming session (using previous session token) in such state will result in some data loss.

class `genai.types.LiveServerSessionResumptionUpdateDict`

Bases: `TypedDict`

Update of the session resumption state.

Skip to content

Only sent if `session_resumption` was set in the connection config.

last_consumed_client_message_index: `Optional[int]`

Index of last message sent by client that is included in state represented by this `SessionResumptionToken`. Only sent when `SessionResumptionConfig.transparent` is set.

Presence of this index allows users to transparently reconnect and avoid issue of losing some part of realtime audio input/video. If client wishes to temporarily disconnect (for example as result of receiving GoAway) they can do it without losing state by buffering messages sent since last `SessionResumptionTokenUpdate`. This field will enable them to limit buffering (avoid keeping all requests in RAM).

Note: This should not be used for when resuming a session at some time later – in those cases partial audio and video frames are likely not needed.

new_handle: `Optional[str]`

New handle that represents state that can be resumed. Empty if `resumable`=false.

resumable: `Optional[bool]`

True if session can be resumed at this point. It might be not possible to resume session at some points. In that case we send update empty new_handle and resumable=false.

Example of such case could be model executing function calls or just generating.

Resuming session (using previous session token) in such state will result in some data loss.

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

FIELDS:

- `session_id (str | None)`

Skip to content `on_id: Optional[str] = None (alias 'sessionId')`
 session id of the live session.

class genai.types.LiveServerSetupCompleteDictBases: `TypedDict`

Sent in response to a `LiveGenerateContentSetup` message from the client.

session_id: `Optional[str]`

The session id of the live session.

pydantic model genai.types.LiveServerToolCallBases: `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.

► 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.LiveServerToolCallCancellationBases: `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.

Only positional-only to allow `self` as a field name.

Skip to content

► 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.

`function_calls`: `Optional[list[FunctionCallDict]]`

The function 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

Skip to content

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

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.

Skip to content [genai.types.LogprobsResultCandidateDict](#)

[... Dict](#)

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

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.

Skip to content

`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.

`self` is explicitly 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`

Skip to content

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`

[Skip to content](#)

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.

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.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')`

Skip to content

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`

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.

`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)`

Skip to content

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.MediaModality(*values)
```

Bases: CaseInsensitiveEnum

Server content modalities.

```
AUDIO = 'AUDIO'
```

Audio.

```
DOCUMENT = 'DOCUMENT'
```

Document, e.g. PDF.

```
IMAGE = 'IMAGE'
```

Images.

```
MODALITY_UNSPECIFIED = 'MODALITY_UNSPECIFIED'
```

The modality is unspecified.

```
TEXT = 'TEXT'
```

Plain text.

```
VIDEO = 'VIDEO'
```

Video.

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

Skip to content

Bases: CaseInsensitiveEnum

The media resolution to use.

MEDIA_RESOLUTION_HIGH = 'MEDIA_RESOLUTION_HIGH'

Media resolution set to high (zoomed reframing with 256 tokens).

MEDIA_RESOLUTION_LOW = 'MEDIA_RESOLUTION_LOW'

Media resolution set to low (64 tokens).

MEDIA_RESOLUTION_MEDIUM = 'MEDIA_RESOLUTION_MEDIUM'

Media resolution set to medium (256 tokens).

MEDIA_RESOLUTION_UNSPECIFIED = 'MEDIA_RESOLUTION_UNSPECIFIED'

Media resolution has not been set

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

Bases: CaseInsensitiveEnum

Server content modalities.

AUDIO = 'AUDIO'

Indicates the model should return audio.

IMAGE = 'IMAGE'

Indicates the model should return images.

MODALITY_UNSPECIFIED = 'MODALITY_UNSPECIFIED'

The modality is unspecified.

TEXT = 'TEXT'

Indicates the model should return text

```
pydantic model genai.types.ModalityTokenCount
```

Bases: BaseModel

Represents token counting info for a single modality.

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 model.

[Skip to content](#)

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

► Show JSON schema

FIELDS:

- `modality` (`genai.types.MediaModality` | `None`)
- `token_count` (`int` | `None`)

field modality: `Optional[MediaModality] = None`

The modality associated with this token count.

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

Number of tokens.

class genai.types.ModalityTokenCountDict

Bases: `TypedDict`

Represents token counting info for a single modality.

modality: `Optional[MediaModality]`

The modality associated with this token count.

token_count: `Optional[int]`

Number of tokens.

class genai.types.Mode(*values)

Bases: `CaseInsensitiveEnum`

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

MODE_DYNAMIC = 'MODE_DYNAMIC'

Run retrieval only when system decides it is necessary.

MODE_UNSPECIFIED = 'MODE_UNSPECIFIED'

Always trigger retrieval.

pydantic model genai.types.Model

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.

Skip to content [.ly](#) positional-only to allow `self` as a field name.

► Show JSON schema

FIELDS:

- checkpoints (list[genai.types.Checkpoint] | None)
- default_checkpoint_id (str | None)
- 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 checkpoints: Optional [list [Checkpoint]] = None

The checkpoints of the model.

field default_checkpoint_id: Optional [str] = None (alias 'defaultCheckpointId')

The default checkpoint id of a model version.

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

Skip to content → name of the model.

field output_token_limit: Optional [int] = None (alias 'outputTokenLimit')

The maximum 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'])`

Skip to content : `list[Part] [Required]`

field role: `Literal['model'] = 'model'`

class genai.types.ModelDict

Bases: `TypedDict`

A trained machine learning model.

checkpoints: `Optional[list[CheckpointDict]]`

The checkpoints of the model.

default_checkpoint_id: `Optional[str]`

The default checkpoint id of a model version.

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]`

Skip to content D 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.ModelSelectionConfig

Bases: `BaseModel`

Config for model selection.

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:

- `feature_selection_preference` (`genai.types.FeatureSelectionPreference` | `None`)

field `feature_selection_preference`: `Optional[FeatureSelectionPreference]` = `None`
`(alias 'featureSelectionPreference')`

Options for feature selection preference.

class genai.types.ModelSelectionConfigDict

Bases: `TypedDict`

Config for model selection.

`feature_selection_preference`: `Optional[FeatureSelectionPreference]`

Options for feature selection preference.

pydantic model genai.types.MultiSpeakerVoiceConfig

Bases: `BaseModel`

The configuration for the multi-speaker setup.

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:

- `speaker_voice_configs` (list[genai.types.SpeakerVoiceConfig] | None)

field speaker_voice_configs: Optional [list [SpeakerVoiceConfig]] = None (alias 'speakerVoiceConfigs')

The configuration for the speaker to use.

class genai.types.MultiSpeakerVoiceConfigDict

Bases: TypedDict

The configuration for the multi-speaker setup.

speaker_voice_configs: Optional [list [SpeakerVoiceConfigDict]]

The configuration for the speaker to use.

class genai.types.Operation

Bases: ABC

A long-running operation.

done: Optional [bool] = FieldInfo(annotation=Union[bool, NoneType], required=False, default=None, description='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]] = FieldInfo(annotation=Union[dict[str, Any], NoneType], required=False, default=None, description='The error result of the operation in case of failure or cancellation.')

abstractmethod classmethod from_api_response(api_response, is_vertex_ai=False)

Creates an Operation from an API response.

RETURN TYPE:

Self

metadata: Optional [dict [str , Any]] = FieldInfo(annotation=Union[dict[str, Any], NoneType], required=False, default=None, description='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] = FieldInfo(annotation=Union[str, NoneType], required=False, default=None, description='The server-assigned name, which is only unique')

Skip to content the same service that originally returns it. If you use the default mapping, the `name` should be a resource name ending with `operations/{unique_id}`.'

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

Bases: CaseInsensitiveEnum

Required. Outcome of the code execution.

```
OUTCOME_DEADLINE_EXCEEDED = 'OUTCOME_DEADLINE_EXCEEDED'
```

Code execution ran for too long, and was cancelled. There may or may not be a partial output present.

```
OUTCOME_FAILED = 'OUTCOME_FAILED'
```

Code execution finished but with a failure. *stderr* should contain the reason.

```
OUTCOME_OK = 'OUTCOME_OK'
```

Code execution completed successfully.

```
OUTCOME_UNSPECIFIED = 'OUTCOME_UNSPECIFIED'
```

Unspecified status. This value should not be used.

```
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 form a valid model.

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

► Show JSON schema

Skip to content

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`)
- `thought_signature` (`bytes | 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).

Skip to content **ht**: `Optional[bool] = None`

Indicates if the part is thought from the model.

```
field thought_signature: Optional[bytes] = None (alias 'thoughtSignature')
```

An opaque signature for the thought so it can be reused in subsequent requests.

```
field video_metadata: Optional[VideoMetadata] = None (alias 'videoMetadata')
```

Metadata for a given video.

```
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=None)
```

Creates a Part from a file uri.

RETURN TYPE:

Part

PARAMETERS:

- **file_uri** (str) – The uri of the file
 - **mime_type** (str) – mime_type: The MIME type of the image. If not provided, the MIME
- Skip to content will be automatically determined.

```
class genai.types.PartDict
```

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.

thought_signature: `Optional[bytes]`

An opaque signature for the thought so it can be reused in subsequent requests.

video_metadata: `Optional[VideoMetadataDict]`

Metadata for a given video.

`genai.types.PartnerModelTuningSpec`

Skip to content

`odel`

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.

`self` is explicitly 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]

Skip to content 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'`

Generate images of adults, but not children.

ALLOW_ALL = `'ALLOW_ALL'`

Generate images that include adults and children.

DONT_ALLOW = `'DONT_ALLOW'`

Block generation of images of people.

pydantic model genai.types.PrebuiltVoiceConfigBases: `BaseModel`

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.PrebuiltVoiceConfigDictBases: `TypedDict`

The configuration for the prebuilt speaker to use.

voice_name: Optional[str]

The name of the prebuilt voice to use.

Skip to content genai.types.ProactivityConfig`.model`

Config for proactivity features.

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:

- `proactive_audio (bool | None)`

field `proactive_audio`: `Optional[bool] = None (alias 'proactiveAudio')`

If enabled, the model can reject responding to the last prompt. For example, this allows the model to ignore out of context speech or to stay silent if the user did not make a request, yet.

class `genai.types.ProactivityConfigDict`

Bases: `TypedDict`

Config for proactivity features.

`proactive_audio: Optional[bool]`

If enabled, the model can reject responding to the last prompt. For example, this allows the model to ignore out of context speech or to stay silent if the user did not make a request, yet.

pydantic model `genai.types.RagChunk`

Bases: `BaseModel`

A RagChunk includes the content of a chunk of a RagFile, and associated metadata.

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:

- `page_span` (`genai.types.RagChunkPageSpan` | `None`)
- `text` (`str` | `None`)

field `page_span`: `Optional[RagChunkPageSpan] = None` (alias '`pageSpan`')

If populated, represents where the chunk starts and ends in the document.

field `text`: `Optional[str] = None`

The content of the chunk.

class `genai.types.RagChunkDict`

Bases: `TypedDict`

A RagChunk includes the content of a chunk of a RagFile, and associated metadata.

`page_span`: `Optional[RagChunkPageSpanDict]`

If populated, represents where the chunk starts and ends in the document.

`text`: `Optional[str]`

The content of the chunk.

pydantic model `genai.types.RagChunkPageSpan`

Bases: `BaseModel`

Represents where the chunk starts and ends in the document.

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:

- `first_page` (`int` | `None`)
- `last_page` (`int` | `None`)

field `first_page`: `Optional[int] = None` (alias '`firstPage`')

Page where chunk starts in the document. Inclusive. 1-indexed.

Skip to content **`page`:** `Optional[int] = None` (alias '`lastPage`')

Page where chunk ends in the document. Inclusive. 1-indexed.

class genai.types.RagChunkPageSpanDictBases: `TypedDict`

Represents where the chunk starts and ends in the document.

`first_page`: `Optional[int]`

Page where chunk starts in the document. Inclusive. 1-indexed.

`last_page`: `Optional[int]`

Page where chunk ends in the document. Inclusive. 1-indexed.

pydantic model genai.types.RagRetrievalConfigBases: `BaseModel`

Specifies the context retrieval 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:

- `filter` (`genai.types.RagRetrievalConfigFilter` | `None`)
- `hybrid_search` (`genai.types.RagRetrievalConfigHybridSearch` | `None`)
- `ranking` (`genai.types.RagRetrievalConfigRanking` | `None`)
- `top_k` (`int` | `None`)

`field filter: Optional[RagRetrievalConfigFilter] = None`

Optional. Config for filters.

`field hybrid_search: Optional[RagRetrievalConfigHybridSearch] = None (alias 'hybridSearch')`

Optional. Config for Hybrid Search.

`field ranking: Optional[RagRetrievalConfigRanking] = None`

Optional. Config for ranking and reranking.

`*: Optional[int] = None (alias 'topK')`

Skip to content

. The number of contexts to retrieve.

class genai.types.RagRetrievalConfigDict

Bases: `TypedDict`

Specifies the context retrieval config.

filter: `Optional[RagRetrievalConfigFilterDict]`

Optional. Config for filters.

hybrid_search: `Optional[RagRetrievalConfigHybridSearchDict]`

Optional. Config for Hybrid Search.

ranking: `Optional[RagRetrievalConfigRankingDict]`

Optional. Config for ranking and reranking.

top_k: `Optional[int]`

Optional. The number of contexts to retrieve.

pydantic model `genai.types.RagRetrievalConfigFilter`

Bases: `BaseModel`

Config for filters.

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:

- `metadata_filter` (`str` | `None`)
- `vector_distance_threshold` (`float` | `None`)
- `vector_similarity_threshold` (`float` | `None`)

field metadata_filter: `Optional[str] = None` (alias '`metadataFilter`')

Optional. String for metadata filtering.

field vector_distance_threshold: `Optional[float] = None` (alias '`vectorDistanceThreshold`')

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

Skip to content **r_similarity_threshold**: `Optional[float] = None` (alias '`SimilarityThreshold`')

Optional. Only returns contexts with vector similarity larger than the threshold.

class genai.types.RagRetrievalConfigFilterDictBases: `TypedDict`

Config for filters.

`metadata_filter`: `Optional[str]`

Optional. String for metadata filtering.

`vector_distance_threshold`: `Optional[float]`

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

`vector_similarity_threshold`: `Optional[float]`

Optional. Only returns contexts with vector similarity larger than the threshold.

pydantic model genai.types.RagRetrievalConfigHybridSearchBases: `BaseModel`

Config for Hybrid Search.

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:

- `alpha` (`float | None`)

`field alpha`: `Optional[float] = None`

Optional. Alpha value controls the weight between dense and sparse vector search results.

The range is $[0, 1]$, while 0 means sparse vector search only and 1 means dense vector search only. The default value is 0.5 which balances sparse and dense vector search equally.

class genai.types.RagRetrievalConfigHybridSearchDictBases: `TypedDict`

Skip to content

Config for Hybrid Search.

alpha: `Optional [float]`

Optional. Alpha value controls the weight between dense and sparse vector search results. The range is [0, 1], while 0 means sparse vector search only and 1 means dense vector search only. The default value is 0.5 which balances sparse and dense vector search equally.

`pydantic model genai.types.RagRetrievalConfigRanking`

Bases: `BaseModel`

Config for ranking and reranking.

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:

- `llm_ranker` (`genai.types.RagRetrievalConfigRankingLlmRanker` | `None`)
- `rank_service` (`genai.types.RagRetrievalConfigRankingRankService` | `None`)

field `llm_ranker`: `Optional [RagRetrievalConfigRankingLlmRanker] = None (alias 'llmRanker')`

Optional. Config for LlmRanker.

field `rank_service`: `Optional [RagRetrievalConfigRankingRankService] = None (alias 'rankService')`

Optional. Config for Rank Service.

`class genai.types.RagRetrievalConfigRankingDict`

Bases: `TypedDict`

Config for ranking and reranking.

`llm_ranker`: `Optional [RagRetrievalConfigRankingLlmRankerDict]`

Optional. Config for LlmRanker.

Skip to content **`e`:** `Optional [RagRetrievalConfigRankingRankServiceDict]`

Optional. Config for Rank Service.

pydantic model genai.types.RagRetrievalConfigRankingLlmRanker

Bases: `BaseModel`

Config for LlmRanker.

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:

- `model_name` (`str` | `None`)

field `model_name`: `Optional[str]` = `None` (alias '`modelName`')

Optional. The model name used for ranking. See [Supported models] (<https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/inference#supported-models>).

class genai.types.RagRetrievalConfigRankingLlmRankerDict

Bases: `TypedDict`

Config for LlmRanker.

`model_name`: `Optional[str]`

//cloud.google.com/vertex-ai/generative-ai/docs/model-reference/inference#supported-models).

TYPE:

Optional. The model name used for ranking. See [Supported models](<https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/inference#supported-models>).

pydantic model genai.types.RagRetrievalConfigRankingRankService

Bases: `BaseModel`

Config for Rank Service.

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:

- `model_name (str | None)`

field `model_name`: Optional [str] = None (alias 'modelName')

Optional. The model name of the rank service. Format: *semantic-ranker-512@latest*

class genai.types.RagRetrievalConfigRankingRankServiceDict

Bases: `TypedDict`

Config for Rank Service.

`model_name: Optional [str]`

semantic-ranker-512@latest

TYPE:

Optional. The model name of the rank service. Format

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 `Dict`

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.RealtimeInputConfig`

Bases: `BaseModel`

Marks the end of user activity.

This can only be sent if automatic (i.e. server-side) activity detection is disabled.

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:

- `activity_handling` (`genai.types.ActivityHandling | None`)
- `automatic_activity_detection` (`genai.types.AutomaticActivityDetection | None`)
- `turn_coverage` (`genai.types.TurnCoverage | None`)

field `activity_handling`: `Optional[ActivityHandling] = None` (alias '`activityHandling`')

Defines what effect activity has.

field `automatic_activity_detection`: `Optional[AutomaticActivityDetection] = None` (alias '`automaticActivityDetection`')

If not set, automatic activity detection is enabled by default. If automatic voice detection is disabled, the client must send activity signals.

Skip to content **coverage**: `Optional[TurnCoverage] = None` (alias '`turnCoverage`')

..... which input is included in the user's turn.

class genai.types.RealtimeInputConfigDict

Bases: TypedDict

Marks the end of user activity.

This can only be sent if automatic (i.e. server-side) activity detection is disabled.

activity_handling: Optional [ActivityHandling]

Defines what effect activity has.

automatic_activity_detection: Optional [AutomaticActivityDetectionDict]

If not set, automatic activity detection is enabled by default. If automatic voice detection is disabled, the client must send activity signals.

turn_coverage: Optional [TurnCoverage]

Defines which input is included in the user's turn.

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.ReplayFileDialogt

Bases: TypedDict

Represents a recorded session.

Skip to content

s: Optional [list [ReplayInteractionDict]]

replay_id: Optional [str]

pydantic model genai.types.ReplayInteraction

Bases: `BaseModel`

Represents a 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

Skip to content

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
```

`class genai.types.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`

[Skip to content](#)

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: `BaseModel`

Defines a retrieval tool that model can call to access external knowledge.

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

Skip to content

`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`)
- `external_api` (`genai.types.ExternalApi` | `None`)
- `vertex_ai_search` (`genai.types.VertexAISeach` | `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 `external_api`: `Optional[ExternalApi]` = `None` (alias 'externalApi')

Use data source powered by external API for grounding.

field `vertex_ai_search`: `Optional[VertexAISeach]` = `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.

pydantic model `genai.types.RetrievalConfig`

Skip to content

Bases: `BaseModel`

Retrieval 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:

- `language_code (str | None)`
- `lat_lng (genai.types.LatLng | None)`

field `language_code`: `Optional[str] = None (alias 'languageCode')`

The language code of the user.

field `lat_lng`: `Optional[LatLng] = None (alias 'latLng')`

Optional. The location of the user.

class `genai.types.RetrievalConfigDict`

Bases: `TypedDict`

Retrieval config.

`language_code`: `Optional[str]`

The language code of the user.

`lat_lng`: `Optional[LatLngDict]`

Optional. The location of the user.

class `genai.types.RetrievalDict`

Skip to content

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.

external_api: `Optional[ExternalApiDict]`

Use data source powered by external API for grounding.

vertex_ai_search: `Optional[VertexAISeachDict]`

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.

`self` is explicitly 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.

Skip to content `genai.types.RetrievalMetadataDict`

----- `ExternalApiDict`

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`)
- `content_type` (`str | None`)
- `scores` (`list[float] | None`)

`field categories: Optional[list[str]] = None`

List of RAI categories.

`field content_type: Optional[str] = None (alias 'contentType')`

Internal use only.

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

List of scores of each categories.

`class genai.types.SafetyAttributesDict`

Skip to content

Bases: `TypedDict`

Safety attributes of a GeneratedImage or the user-provided prompt.

categories: `Optional[list[str]]`

List of RAI categories.

content_type: `Optional[str]`

Internal use only.

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`

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

Skip to content

FIELDS:

- `blocked` (`bool` | `None`)
- `category` (`genai.types.HarmCategory` | `None`)
- `overwritten_threshold` (`genai.types.HarmBlockThreshold` | `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 `overwritten_threshold`: `Optional[HarmBlockThreshold] = None (alias 'overwrittenThreshold')`

Output only. The overwritten threshold for the safety category of Gemini 2.0 image out. If minors are detected in the output image, the threshold of each safety category will be overwritten if user sets a lower threshold.

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.

overwritten_threshold: `Optional[HarmBlockThreshold]`

Output only. The overwritten threshold for the safety category of Gemini 2.0 image out. If minors are detected in the output image, the threshold of each safety category will be overwritten if user sets a lower threshold.

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

Skip to content

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.

field threshold: `Optional[HarmBlockThreshold] = None`

Required. The harm block threshold.

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.

class genai.types.Scale(*values)

[Skip to content](#)

Bases: `CaseInsensitiveEnum`

Scale of the generated music.

`A_FLAT_MAJOR_F_MINOR` = `'A_FLAT_MAJOR_F_MINOR'`

Ab major or F minor.

`A_MAJOR_G_FLAT_MINOR` = `'A_MAJOR_G_FLAT_MINOR'`

A major or Gb minor.

`B_FLAT_MAJOR_G_MINOR` = `'B_FLAT_MAJOR_G_MINOR'`

Bb major or G minor.

`B_MAJOR_A_FLAT_MINOR` = `'B_MAJOR_A_FLAT_MINOR'`

B major or Ab minor.

`C_MAJOR_A_MINOR` = `'C_MAJOR_A_MINOR'`

C major or A minor.

`D_FLAT_MAJOR_B_FLAT_MINOR` = `'D_FLAT_MAJOR_B_FLAT_MINOR'`

Db major or Bb minor.

`D_MAJOR_B_MINOR` = `'D_MAJOR_B_MINOR'`

D major or B minor.

`E_FLAT_MAJOR_C_MINOR` = `'E_FLAT_MAJOR_C_MINOR'`

Eb major or C minor

`E_MAJOR_D_FLAT_MINOR` = `'E_MAJOR_D_FLAT_MINOR'`

E major or Db minor.

`F_MAJOR_D_MINOR` = `'F_MAJOR_D_MINOR'`

F major or D minor.

`G_FLAT_MAJOR_E_FLAT_MINOR` = `'G_FLAT_MAJOR_E_FLAT_MINOR'`

Gb major or Eb minor.

`G_MAJOR_E_MINOR` = `'G_MAJOR_E_MINOR'`

G major or E minor.

`UNSPECIFIED` = `'SCALE_UNSPECIFIED'`

Skip to content value. This value is unused.

`pydantic model genai.types.Schema`

Bases: [BaseModel](#)

Schema is used to define the format of input/output data.

Represents a select subset of an [OpenAPI 3.0 schema object](<https://spec.openapis.org/oas/v3.0.3#schema-object>). More fields may be added in the future as needed.

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:

- `additional_properties` (`Any | None`)
- `any_of` (`list[genai.types.Schema] | None`)
- `default` (`Any | None`)
- `defs` (`dict[str, genai.types.Schema] | 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`)
- `ref` (`str | None`)
- `required` (`list[str] | None`)
- `title` (`str | None`)
- `type` (`genai.types.Type | None`)

field `additional_properties`: `Optional[Any] = None (alias 'additionalProperties')`

Optional. Can either be a boolean or an object; controls the presence of additional properties.

field `any_of`: `Optional[list[Schema]] = None (alias 'anyOf')`

Optional. The value should be validated against any (one or more) of the subschemas in

..

Skip to content

default: `Optional[Any] = None`

Optional. Default value of the data.

field `defs`: `Optional[dict[str, Schema]] = None`

Optional. A map of definitions for use by `ref` Only allowed at the root of the schema.

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"]}

field `example`: `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

Skip to content **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.

field ref: Optional[str] = None

Optional. Allows indirect references between schema nodes. The value should be a valid reference to a child of the root *defs*. For example, the following schema defines a reference to a schema node named "Pet": type: object properties: pet: ref: #/defs/Pet defs: Pet: type: object properties: name: type: string The value of the "pet" property is a reference to the schema node named "Pet". See details in <https://json-schema.org/understanding-json-schema/structuring>

field required: 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.

classmethod from_json_schema(*, json_schema, api_option='GEMINI_API', raise_error_on_unsupported_field=False)

Converts a JSONSchema object to a Schema object.

Skip to content

The JSONSchema is compatible with 2020-12 JSON Schema draft, specified by OpenAPI 3.1.

RETURN TYPE:

Schema

PARAMETERS:

- **json_schema** – JSONSchema object to be converted.
- **api_option** – API option to be used. If set to ‘VERTEX_AI’, the JSONSchema will be converted to a Schema object that is compatible with Vertex AI API. If set to ‘GEMINI_API’, the JSONSchema will be converted to a Schema object that is compatible with Gemini API. Default is ‘GEMINI_API’.
- **raise_error_on_unsupported_field** – If set to True, an error will be raised if the JSONSchema contains any unsupported fields. Default is False.

RETURNS:

Schema object that is compatible with the specified API option.

RAISES:

ValueError – If the JSONSchema contains any unsupported fields and raise_error_on_unsupported_field is set to True. Or if the JSONSchema is not compatible with the specified API option.

property **json_schema**: JSONSchema

Converts the Schema object to a JSONSchema object, that is compatible with 2020-12 JSON Schema draft.

If a Schema field is not supported by JSONSchema, it will be ignored.

class **genai.types.SchemaDict**

Skip to content

Bases: `TypedDict`

Schema is used to define the format of input/output data.

Represents a select subset of an [OpenAPI 3.0 schema object]

(<https://spec.openapis.org/oas/v3.0.3#schema-object>). More fields may be added in the future as needed.

additional_properties: `Optional[Any]`

Optional. Can either be a boolean or an object; controls the presence of additional properties.

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.

defs: `Optional[dict[str, SchemaDict]]`

Optional. A map of definitions for use by `ref` Only allowed at the root of the schema.

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]`

Skip to content

“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.

maximum: Optional [float]

Optional. 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.

Skip to content

dering: 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.

ref: Optional [str]

object properties: pet: ref: #/defs/Pet defs: Pet: type: object properties: name: type: string
The value of the “pet” property is a reference to the schema node named “Pet”. See details in <https://json-schema.org/understanding-json-schema/structuring>

TYPE:

Optional. Allows indirect references between schema nodes. The value should be a valid reference to a child of the root *defs*. For example, the following schema defines a reference to a schema node named “Pet”

TYPE:

type

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

FIELDS:

- `rendered_content` (str | None)
- `sdk_blob` (bytes | None)

field rendered_content: Optional [str] = None (alias 'renderedContent')

Web content snippet that can be embedded in a web page or an app webview.

Skip to content

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

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.

Skip to content

`Optional[str] = None`

Output only. The text corresponding to the segment from the response.

class genai.types.SegmentDict

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.SessionResumptionConfig

Skip to content

Bases: `BaseModel`

Configuration of session resumption mechanism.

Included in `LiveConnectConfig.session_resumption`. If included server will send `LiveServerSessionResumptionUpdate` 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:

- `handle (str | None)`
- `transparent (bool | None)`

field `handle`: `Optional [str] = None`

Session resumption handle of previous session (session to restore).

If not present new session will be started.

field `transparent`: `Optional [bool] = None`

If set the server will send `last_consumed_client_message_index` in the `session_resumption_update` messages to allow for transparent reconnections.

class `genai.types.SessionResumptionConfigDict`

Bases: `TypedDict`

Configuration of session resumption mechanism.

Included in `LiveConnectConfig.session_resumption`. If included server will send `LiveServerSessionResumptionUpdate` messages.

`handle`: `Optional [str]`

Session resumption handle of previous session (session to restore).

If not present new session will be started.

`transparent`: `Optional [bool]`

Skip to content server will send `last_consumed_client_message_index` in the `resumption_update` messages to allow for transparent reconnections.

pydantic model `genai.types.SlidingWindow`

Bases: `BaseModel`

Context window will be truncated by keeping only suffix of it.

Context window will always be cut at start of USER role turn. System instructions and `BidiGenerateContentSetup.prefix_turns` will not be subject to the sliding window mechanism, they will always stay at the beginning of context window.

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:

- `target_tokens (int | None)`

field `target_tokens`: `Optional[int] = None (alias 'targetTokens')`

Session reduction target – how many tokens we should keep. Window shortening operation has some latency costs, so we should avoid running it on every turn. Should be < `trigger_tokens`. If not set, `trigger_tokens/2` is assumed.

class `genai.types.SlidingWindowDict`

Bases: `TypedDict`

Context window will be truncated by keeping only suffix of it.

Context window will always be cut at start of USER role turn. System instructions and `BidiGenerateContentSetup.prefix_turns` will not be subject to the sliding window mechanism, they will always stay at the beginning of context window.

`target_tokens`: `Optional[int]`

Session reduction target – how many tokens we should keep. Window shortening operation has some latency costs, so we should avoid running it on every turn. Should be < `trigger_tokens`. If not set, `trigger_tokens/2` is assumed.

pydantic model `genai.types.SpeakerVoiceConfig`

Bases: `BaseModel`

The configuration for the speaker to use.

Skip to 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:

- `speaker` (str | None)
- `voice_config` (genai.types.VoiceConfig | None)

field speaker: `Optional[str] = None`

The name of the speaker to use. Should be the same as in the prompt.

field voice_config: `Optional[VoiceConfig] = None (alias 'voiceConfig')`

The configuration for the voice to use.

class genai.types.SpeakerVoiceConfigDict

Bases: `TypedDict`

The configuration for the speaker to use.

speaker: `Optional[str]`

The name of the speaker to use. Should be the same as in the prompt.

voice_config: `Optional[VoiceConfigDict]`

The configuration for the voice to use.

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

Skip to content

FIELDS:

- `language_code (str | None)`
- `multi SpeakerVoiceConfig (genai.types.MultiSpeakerVoiceConfig | None)`
- `voice_config (genai.types.VoiceConfig | None)`

field language_code: `Optional[str] = None (alias 'languageCode')`

Language code (ISO 639, e.g. en-US) for the speech synthesis. Only available for Live API.

field multi SpeakerVoiceConfig: `Optional[MultiSpeakerVoiceConfig] = None (alias 'multiSpeakerVoiceConfig')`

The configuration for the multi-speaker setup. It is mutually exclusive with the voice_config field.

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.

language_code: `Optional[str]`

Language code (ISO 639, e.g. en-US) for the speech synthesis. Only available for Live API.

multi SpeakerVoiceConfig: `Optional[MultiSpeakerVoiceConfigDict]`

The configuration for the multi-speaker setup. It is mutually exclusive with the voice_config field.

voice_config: `Optional[VoiceConfigDict]`

The configuration for the speaker to use.

class genai.types.StartSensitivity(*values)

Skip to content

Bases: CaseInsensitiveEnum

Start of speech sensitivity.

START_SENSITIVITY_HIGH = 'START_SENSITIVITY_HIGH'

Automatic detection will detect the start of speech more often.

START_SENSITIVITY_LOW = 'START_SENSITIVITY_LOW'

Automatic detection will detect the start of speech less often.

START_SENSITIVITY_UNSPECIFIED = 'START_SENSITIVITY_UNSPECIFIED'

The default is START_SENSITIVITY_LOW.

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

Skip to content

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`

Skip to content

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`

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.

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.

Skip to content

↓ 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]`

The subject type of a subject reference image.

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

Skip to content

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.

VALIDATED BY:

- `_validate_mask_image_config`

field subject_image_config: `Optional[SubjectReferenceConfig] = None (alias 'subjectImageConfig')`

VALIDATED BY:

- `_validate_mask_image_config`

Skip to content [yes](#).**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.

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

► Show JSON schema

Skip to content

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. Mutually exclusive with `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. Mutually exclusive with `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.

Skip to content `NotFoundError][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:

- `dropped_example_reasons` (`list[str] | None`)
- `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 `dropped_example_reasons`: `Optional[list[str]] = None (alias 'droppedExampleReasons')`

Output only. For each index in `truncated_example_indices`, the user-facing reason why the example was dropped.

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')`

Output only. The number of examples in the dataset that have been dropped. An example can be dropped for reasons including: too many tokens, contains an invalid image, contains too many images, etc.

Skip to content **`_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')
```

Output only. A partial sample of the indices (starting from 1) of the dropped 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.

```
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[ 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.

dropped_example_reasons: `Optional [list [str]]`

Output only. For each index in `truncated_example_indices`, the user-facing reason why the example was dropped.

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]`

too many tokens, contains an invalid image, contains too many images, etc.

TYPE:

Output only. The number of examples in the dataset that have been dropped. An example can be dropped for reasons including

total_tuning_character_count: `Optional [int]`

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

truncated_example_indices: `Optional [list [int]]`

Output only. A partial sample of the indices (starting from 1) of the dropped 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]`

Skip to content 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

[Skip to content](#)

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.

`self` is explicitly 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.

Skip to content `Optional[float] = None`

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.

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 (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

Skip to content

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

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.

Skip to content

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

FIELDS:

- `export_last_checkpoint_only` (`bool | None`)
- `hyper_parameters` (`genai.types.SupervisedHyperParameters | None`)
- `training_dataset_uri` (`str | None`)
- `validation_dataset_uri` (`str | None`)

field `export_last_checkpoint_only`: `Optional[bool] = None` (alias '`exportLastCheckpointOnly`')

Optional. If set to true, disable intermediate checkpoints for SFT and only the last checkpoint will be exported. Otherwise, enable intermediate checkpoints for SFT. Default is false.

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

Optional. Hyperparameters for SFT.

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

Required. Training dataset used for tuning. The dataset can be specified as either a Cloud Storage path to a JSONL file or as the resource name of a Vertex Multimodal Dataset.

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

Optional. Validation dataset used for tuning. The dataset can be specified as either a Cloud Storage path to a JSONL file or as the resource name of a Vertex Multimodal Dataset.

class `genai.types.SupervisedTuningSpecDict`

– `Dict`

Skip to content

Tuning Spec for Supervised Tuning for first party models.

export_last_checkpoint_only: `Optional[bool]`

Optional. If set to true, disable intermediate checkpoints for SFT and only the last checkpoint will be exported. Otherwise, enable intermediate checkpoints for SFT. Default is false.

hyper_parameters: `Optional[SupervisedHyperParametersDict]`

Optional. Hyperparameters for SFT.

training_dataset_uri: `Optional[str]`

Required. Training dataset used for tuning. The dataset can be specified as either a Cloud Storage path to a JSONL file or as the resource name of a Vertex Multimodal Dataset.

validation_dataset_uri: `Optional[str]`

Optional. Validation dataset used for tuning. The dataset can be specified as either a Cloud Storage path to a JSONL file or as the resource name of a Vertex Multimodal Dataset.

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

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 +---+ .table: `Optional[list[TestTableItem]] = None (alias 'testTable')`

Skip to content

----- `...pes.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`)
- `ignore_keys` (`list[str] | 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 ignore_keys: `Optional[list[str]] = None` (alias 'ignoreKeys')

Keys to ignore when comparing the request and response. This is useful for tests that are not deterministic.

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')

If you don't want to use the default replay id which is derived from the test name.

Skip to content

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

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.

ignore_keys: Optional[list[str]]

Keys to ignore when comparing the request and response. This is useful for tests that are not deterministic.

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.

^ --> `` model by parsing and validating input data from keyword arguments.

Skip to content

`actionError][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)`
- `thinking_budget (int | 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.

field `thinking_budget`: `Optional[int] = None (alias 'thinkingBudget')`

Indicates the thinking budget in tokens. 0 is DISABLED. -1 is AUTOMATIC. The default values and allowed ranges are model dependent.

class `genai.types.ThinkingConfigDict`

Bases: `TypedDict`

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.

`thinking_budget`: `Optional[int]`

Indicates the thinking budget in tokens. 0 is DISABLED. -1 is AUTOMATIC. The default values and allowed ranges are model dependent.

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

Skip to content

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.

pydantic model `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

Skip to content

FIELDS:

- `code_execution` (`genai.types.ToolCodeExecution` | `None`)
- `computer_use` (`genai.types.ToolComputerUse` | `None`)
- `enterprise_web_search` (`genai.types.EnterpriseWebSearch` | `None`)
- `function_declarations` (`list[genai.types.FunctionDeclaration]` | `None`)
- `google_maps` (`genai.types.GoogleMaps` | `None`)
- `google_search` (`genai.types.GoogleSearch` | `None`)
- `google_search_retrieval` (`genai.types.GoogleSearchRetrieval` | `None`)
- `retrieval` (`genai.types.Retrieval` | `None`)
- `url_context` (`genai.types.UrlContext` | `None`)

field `code_execution`: `Optional[ToolCodeExecution] = None (alias 'codeExecution')`

Optional. CodeExecution tool type. Enables the model to execute code as part of generation.

field `computer_use`: `Optional[ToolComputerUse] = None (alias 'computerUse')`

Optional. Tool to support the model interacting directly with the computer. If enabled, it automatically populates computer-use specific Function Declarations.

field `enterprise_web_search`: `Optional[EnterpriseWebSearch] = None (alias 'enterpriseWebSearch')`

Optional. Enterprise web search tool type. Specialized retrieval tool that is powered by Vertex AI Search and Sec4 compliance.

field `function_declarations`: `Optional[list[FunctionDeclaration]] = None (alias 'functionDeclarations')`

List of function declarations that the tool supports.

field `google_maps`: `Optional[GoogleMaps] = None (alias 'googleMaps')`

Optional. Google Maps tool type. Specialized retrieval tool that is powered by Google Maps.

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

Skip to content 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.

field url_context: Optional[UrlContext] = None (alias 'urlContext')

Optional. Tool to support URL context retrieval.

pydantic model genai.types.ToolCodeExecution

Bases: `BaseModel`

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.

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.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.ToolComputerUse

Bases: `BaseModel`

Tool to support computer 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.

^N schema

Skip to content

FIELDS:

- `environment (genai.types.Environment | None)`

field environment: `Optional[Environment] = None`

Required. The environment being operated.

`class genai.types.ToolComputerUseDict`

Bases: `TypedDict`

Tool to support computer use.

environment: `Optional[Environment]`

Required. The environment being operated.

`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)`
- `retrieval_config (genai.types.RetrievalConfig | None)`

field function_calling_config: `Optional[FunctionCallingConfig] = None (alias 'functionCallingConfig')`

Optional. Function calling config.

field retrieval_config: `Optional[RetrievalConfig] = None (alias 'retrievalConfig')`

Optional. Retrieval config.

`class genai.types.ToolConfigDict`

Skip to content `Dict`

Tool config.

This config is shared for all tools provided in the request.

function_calling_config: Optional [`FunctionCallingConfigDict`]

Optional. Function calling config.

retrieval_config: Optional [`RetrievalConfigDict`]

Optional. Retrieval config.

class `genai.types.ToolDict`

[Skip to content](#)

Bases: `TypedDict`

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.

computer_use: `Optional[ToolComputerUseDict]`

Optional. Tool to support the model interacting directly with the computer. If enabled, it automatically populates computer-use specific Function Declarations.

enterprise_web_search: `Optional[EnterpriseWebSearchDict]`

Optional. Enterprise web search tool type. Specialized retrieval tool that is powered by Vertex AI Search and Sec4 compliance.

function_declarations: `Optional[list[FunctionDeclarationDict]]`

List of function declarations that the tool supports.

google_maps: `Optional[GoogleMapsDict]`

Optional. Google Maps tool type. Specialized retrieval tool that is powered by Google Maps.

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.

url_context: `Optional[UrlContextDict]`

Optional. Tool to support URL context retrieval.

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

`~nSensitiveEnum`

Skip to content

Traffic type. This shows whether a request consumes Pay-As-You-Go or Provisioned Throughput quota.

ON_DEMAND = 'ON_DEMAND'

Type for Pay-As-You-Go traffic.

PROVISIONED_THROUGHPUT = 'PROVISIONED_THROUGHPUT'

Type for Provisioned Throughput traffic.

TRAFFIC_TYPE_UNSPECIFIED = 'TRAFFIC_TYPE_UNSPECIFIED'

Unspecified request traffic type.

pydantic model genai.types.Transcription

Bases: `BaseModel`

Audio transcription in Server 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:

- `finished` (`bool` | `None`)
- `text` (`str` | `None`)

field finished: `Optional[bool]` = `None`

The bool indicates the end of the transcription.

field text: `Optional[str]` = `None`

Transcription text.

class genai.types.TranscriptionDict

Skip to content

Bases: `TypedDict`

Audio transcription in Server Conent.

finished: `Optional[bool]`

The bool indicates the end of the transcription.

text: `Optional[str]`

Transcription text.

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:

- `checkpoints` (`list[genai.types.TunedModelCheckpoint] | None`)
- `endpoint` (`str | None`)
- `model` (`str | None`)

field checkpoints: `Optional[list[TunedModelCheckpoint]] = None`

The checkpoints associated with this TunedModel. This field is only populated for tuning jobs that enable intermediate checkpoints.

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}`.

pydantic model `genai.types.TunedModelCheckpoint`

Bases: `BaseModel`

Skip to content ↗checkpoint for the Tuned Model of 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

FIELDS:

- `checkpoint_id` (str | None)
- `endpoint` (str | None)
- `epoch` (int | None)
- `step` (int | None)

field `checkpoint_id`: Optional[str] = None (alias 'checkpointId')

The ID of the checkpoint.

field `endpoint`: Optional[str] = None

The Endpoint resource name that the checkpoint is deployed to. Format:
projects/{project}/locations/{location}/endpoints/{endpoint}.

field `epoch`: Optional[int] = None

The epoch of the checkpoint.

field `step`: Optional[int] = None

The step of the checkpoint.

class `genai.types.TunedModelCheckpointDict`

Bases: `TypedDict`

TunedModelCheckpoint for the Tuned Model of a Tuning Job.

`checkpoint_id`: Optional[str]

The ID of the checkpoint.

`endpoint`: Optional[str]

The Endpoint resource name that the checkpoint is deployed to. Format:
projects/{project}/locations/{location}/endpoints/{endpoint}.

`epoch`: Optional[int]

The epoch of the checkpoint.

Skip to content `Optional[int]`

The step of the checkpoint.

class genai.types.TunedModelDict

Bases: `TypedDict`

checkpoints: `Optional [list [TunedModelCheckpointDict]]`

The checkpoints associated with this TunedModel. This field is only populated for tuning jobs that enable intermediate checkpoints.

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

[Skip to content](#)

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`

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`

Skip to content `odel`

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`]

The SFT 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.

Skip to content

Show schema

FIELDS:

- `examples` (`list[genai.types.TuningExample] | None`)
- `gcs_uri` (`str | None`)
- `vertex_dataset_resource` (`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.

field vertex_dataset_resource: `Optional[str] = None (alias 'vertexDatasetResource')`

The resource name of the Vertex Multimodal Dataset that is used as training dataset.

Example: 'projects/my-project-id-or-number/locations/my-location/datasets/my-dataset-id'.

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.

vertex_dataset_resource: `Optional[str]`

'projects/my-project-id-or-number/locations/my-location/datasets/my-dataset-id'.

TYPE:

The resource name of the Vertex Multimodal Dataset that is used as training dataset.

Example

pydantic model genai.types.TuningExample

Bases: `BaseModel`

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

Skip to content `ValidationError][pydantic_core.ValidationError]` if the input data cannot be validated to 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)`
- `satisfies_pzi (bool | None)`
- `satisfies_pzs (bool | None)`
- `service_account (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. See [Supported models]
https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/tuning#supported_models.

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.

Skip to content **`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`.

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 satisfies_pzi: Optional [`bool`] = None (alias 'satisfiesPzi')

Output only. Reserved for future use.

field satisfies_pzs: Optional [`bool`] = None (alias 'satisfiesPzs')

Skip to content only. Reserved for future use.

field service_account: Optional [`str`] = None (alias 'serviceAccount')

The service account that the tuningJob workload runs as. If not specified, the Vertex AI Secure Fine-Tuned Service Agent in the project will be used. See <https://cloud.google.com/iam/docs/service-agents#vertex-ai-secure-fine-tuning-service-agent>. Users starting the pipeline must have the `iam.serviceAccounts.actAs` permission on this service account.

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')`

Output only. The tuning data statistics associated with this TuningJob.

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`

Bases: `TypedDict`

Skip to content

A tuning job.

base_model: `Optional [str]`

//cloud.google.com/vertex-ai/generative-ai/docs/model-reference/tuning#supported_models).

TYPE:

The base model that is being tuned. See [Supported models](https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/tuning#supported_models)

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 Skip to content 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://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/tuning#supported_labels

name: Optional [str]

projects/{project}/locations/{location}/tuningJobs/{tuning_job}

TYPE:

Output only. Identifier. Resource name of a TuningJob. Format

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

satisfies_pzi: Optional [bool]

Output only. Reserved for future use.

satisfies_pzs: Optional [bool]

Output only. Reserved for future use.

service_account: Optional [str]

//cloud.google.com/iam/docs/service-agents#vertex-ai-secure-fine-tuning-service-agent
Users starting the pipeline must have the *iam.serviceAccounts.actAs* permission on this service account.

TYPE:

The service account that the tuningJob workload runs as. If not specified, the Vertex AI Secure Fine-Tuned Service Agent in the project will be used. See https

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.

Skip to content : 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)
- vertex_dataset_resource (str | None)

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

GCS URI of the file containing validation dataset in JSONL format.

field vertex_dataset_resource: Optional [str] = None (alias 'vertexDatasetResource')

The resource name of the Vertex Multimodal Dataset that is used as training dataset.

Example: 'projects/my-project-id-or-number/locations/my-location/datasets/my-dataset-id'.

class genai.types.TuningValidationDatasetDict

Skip to content

Bases: `TypedDict`

gcs_uri: `Optional [str]`

GCS URI of the file containing validation dataset in JSONL format.

vertex_dataset_resource: `Optional [str]`

'projects/my-project-id-or-number/locations/my-location/datasets/my-dataset-id'.

TYPE:

The resource name of the Vertex Multimodal Dataset that is used as training dataset.

Example

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

Bases: `CaseInsensitiveEnum`

Options about which input is included in the user's turn.

TURN_COVERAGE_UNSPECIFIED = `'TURN_COVERAGE_UNSPECIFIED'`

If unspecified, the default behavior is `TURN_INCLUDES_ONLY_ACTIVITY`.

TURN_INCLUDES_ALL_INPUT = `'TURN_INCLUDES_ALL_INPUT'`

The users turn includes all realtime input since the last turn, including inactivity (e.g. silence on the audio stream).

TURN_INCLUDES_ONLY_ACTIVITY = `'TURN_INCLUDES_ONLY_ACTIVITY'`

The users turn only includes activity since the last turn, excluding inactivity (e.g. silence on the audio stream). This is the default behavior.

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

Skip to content

Bases: `CaseInsensitiveEnum`

Optional. The type of the data.

ARRAY = `'ARRAY'`

OpenAPI array type

BOOLEAN = `'BOOLEAN'`

OpenAPI boolean type

INTEGER = `'INTEGER'`

OpenAPI integer type

NULL = `'NULL'`

Null type

NUMBER = `'NUMBER'`

OpenAPI number type

OBJECT = `'OBJECT'`

OpenAPI object type

STRING = `'STRING'`

OpenAPI string type

TYPE_UNSPECIFIED = `'TYPE_UNSPECIFIED'`

Not specified, should not be used.

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

Skip to content

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. Uses RFC 3339 format, Example:
2014-10-02T15:01:23Z.

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. It is a duration string, with up to nine fractional digits, terminated by 's'. Example: "3.5s".

class `genai.types.UpdateCachedContentConfigDict`

Bases: `TypedDict`

Optional parameters for caches.update method.

`expire_time`: `Optional[datetime]`

01:23Z.

TYPE:

Timestamp of when this resource is considered expired. Uses RFC 3339 format,
Example

TYPE:

2014-10-02T15

`http_options`: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

`ttl`: `Optional[str]`

"3.5s".

TYPE:

The TTL for this resource. The expiration time is computed

TVDC.

Skip to content + TTL. It is a duration string, with up to nine fractional digits, terminated by 's'.

Example

pydantic model genai.types.UpdateModelConfigBases: `BaseModel`

Configuration for updating a tuned 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:

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

`field default_checkpoint_id: Optional[str] = None (alias 'defaultCheckpointId')`

`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.UpdateModelConfigDictBases: `TypedDict`

Configuration for updating a tuned model.

`default_checkpoint_id: Optional[str]`

`description: Optional[str]`

`display_name: Optional[str]`

`http_options: Optional[HttpOptionsDict]`

Used to override HTTP request options.

pydantic model genai.types.UploadFileConfigBases: `BaseModel`

Skip to content → ide 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

Skip to content

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`

name: `Optional[str]`

The name 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

Skip to content

FIELDS:

- `enhance_input_image (bool | None)`
- `http_options (genai.types.HttpOptions | None)`
- `image_preservation_factor (float | None)`
- `include_rai_reason (bool | None)`
- `output_compression_quality (int | None)`
- `output_mime_type (str | None)`

field `enhance_input_image`: `Optional[bool] = None (alias 'enhanceInputImage')`

Whether to add an image enhancing step before upscaling. It is expected to suppress the noise and JPEG compression artifacts from the input image.

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

Used to override HTTP request options.

field `image_preservation_factor`: `Optional[float] = None (alias 'imagePreservationFactor')`

With a higher image preservation factor, the original image pixels are more respected. With a lower image preservation factor, the output image will have be more different from the input image, but with finer details and less noise.

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](#).

enhance_input_image: `Optional[bool]`

Whether to add an image enhancing step before upscaling. It is expected to suppress the noise and JPEG compression artifacts from the input image.

http_options: `Optional[HttpOptionsDict]`

Used to override HTTP request options.

image_preservation_factor: `Optional[float]`

With a higher image preservation factor, the original image pixels are more respected. With a lower image preservation factor, the output image will have be more different from the input image, but with finer details and less noise.

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

Skip to content

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.

field upscale_factor: `Optional[str] = None (alias 'upscaleFactor')`

The factor 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.

Skip to content `NotFoundError][pydantic_core.ValidationError]` if the input data cannot be validated to `UpscaleImageParameters` 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.UrlContext

Bases: BaseModel

Tool to support URL context 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

class genai.types.UrlContextDict

Bases: TypedDict

Tool to support URL context retrieval.

pydantic model genai.types.UrlContextMetadata

Bases: BaseModel

Metadata related to url context retrieval tool.

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:

- `url_metadata (list[genai.types.UrlMetadata] | None)`

field `url_metadata`: `Optional[list[UrlMetadata]] = None (alias 'urlMetadata')`

List of url context.

class `genai.types.UrlContextMetadataDict`

Bases: `TypedDict`

Metadata related to url context retrieval tool.

url_metadata: `Optional[list[UrlMetadataDict]]`

List of url context.

pydantic model `genai.types.UrlMetadata`

Bases: `BaseModel`

Context for a single url 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:

- `retrieved_url (str | None)`
- `url_retrieval_status (genai.types.UrlRetrievalStatus | None)`

field `retrieved_url`: `Optional[str] = None (alias 'retrievedUrl')`

The URL retrieved by the tool.

field `url_retrieval_status`: `Optional[UrlRetrievalStatus] = None (alias 'urlRetrievalStatus')`

Status of the url retrieval.

class `genai.types.UrlMetadataDict`

Bases: `TypedDict`

Skip to content

Context for a single url retrieval.

retrieved_url: `Optional[str]`

The URL retrieved by the tool.

url_retrieval_status: `Optional[UrlRetrievalStatus]`

Status of the url retrieval.

class genai.types.UrlRetrievalStatus(*values)

Bases: `CaseInsensitiveEnum`

Status of the url retrieval.

URL_RETRIEVAL_STATUS_ERROR = 'URL_RETRIEVAL_STATUS_ERROR'

Url retrieval is failed due to error.

URL_RETRIEVAL_STATUS_SUCCESS = 'URL_RETRIEVAL_STATUS_SUCCESS'

Url retrieval is successful.

URL_RETRIEVAL_STATUS_UNSPECIFIED = 'URL_RETRIEVAL_STATUS_UNSPECIFIED'

Default value. This value is unused

pydantic model genai.types.UsageMetadata

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:

- `cache_tokens_details` (`list[genai.types.ModalityTokenCount] | None`)
- `cached_content_token_count` (`int | None`)
- `prompt_token_count` (`int | None`)
- `prompt_tokens_details` (`list[genai.types.ModalityTokenCount] | None`)
- `response_token_count` (`int | None`)
- `response_tokens_details` (`list[genai.types.ModalityTokenCount] | None`)
- `thoughts_token_count` (`int | None`)
- `tool_use_prompt_token_count` (`int | None`)
- `tool_use_prompt_tokens_details` (`list[genai.types.ModalityTokenCount] | None`)
- `total_token_count` (`int | None`)
- `traffic_type` (`genai.types.TrafficType | None`)

field `cache_tokens_details`: `Optional[list[ModalityTokenCount]] = None` (alias '`cacheTokensDetails`')

List of modalities that were processed in the cache input.

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

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

field `prompt_token_count`: `Optional[int] = None` (alias '`promptTokenCount`')

Number of tokens in the prompt. 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 `prompt_tokens_details`: `Optional[list[ModalityTokenCount]] = None` (alias '`promptTokensDetails`')

List of modalities that were processed in the request input.

field `response_token_count`: `Optional[int] = None` (alias '`responseTokenCount`')

Total number of tokens across all the generated response candidates.

field `response_tokens_details`: `Optional[list[ModalityTokenCount]] = None` (alias '`responseTokensDetails`')

List of modalities that were returned in the response.

field `thoughts_token_count`: `Optional[int] = None` (alias '`thoughtsTokenCount`')

of tokens of thoughts for thinking models.

Skip to content

field `tool_use_prompt_token_count`: `Optional[int] = None` (alias '`toolUsePromptTokenCount`')

Number of tokens present in tool-use prompt(s).

```
field tool_use_prompt_tokens_details: Optional [ list [ ModalityTokenCount ] ] = None  
    (alias 'toolUsePromptTokensDetails')
```

List of modalities that were processed in the tool-use prompt.

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

Total token count for prompt, response candidates, and tool-use prompts(if present).

```
field traffic_type: Optional [ TrafficType ] = None (alias 'trafficType')
```

Traffic type. This shows whether a request consumes Pay-As-You-Go or Provisioned Throughput quota.

```
class genai.types.UsageMetadataDict
```

[Skip to content](#)

Bases: `TypedDict`

Usage metadata about response(s).

cache_tokens_details: `Optional [list [ModalityTokenCountDict]]`

List of modalities that were processed in the cache input.

cached_content_token_count: `Optional [int]`

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

prompt_token_count: `Optional [int]`

Number of tokens in the prompt. When `cached_content` is set, this is still the total effective prompt size meaning this includes the number of tokens in the cached content.

prompt_tokens_details: `Optional [list [ModalityTokenCountDict]]`

List of modalities that were processed in the request input.

response_token_count: `Optional [int]`

Total number of tokens across all the generated response candidates.

response_tokens_details: `Optional [list [ModalityTokenCountDict]]`

List of modalities that were returned in the response.

thoughts_token_count: `Optional [int]`

Number of tokens of thoughts for thinking models.

tool_use_prompt_token_count: `Optional [int]`

Number of tokens present in tool-use prompt(s).

tool_use_prompt_tokens_details: `Optional [list [ModalityTokenCountDict]]`

List of modalities that were processed in the tool-use prompt.

total_token_count: `Optional [int]`

Total token count for prompt, response candidates, and tool-use prompts(if present).

traffic_type: `Optional [TrafficType]`

Traffic type. This shows whether a request consumes Pay-As-You-Go or Provisioned Throughput quota.

pydantic model `genai.types.UserContent`

Skip to content `nt`

UserContent facilitates the creation of a Content object with a user role.

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_uri="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.VertexAISeach

Bases: `BaseModel`

Retrieve from Vertex AI Search datastore or engine for grounding.

datastore and engine are mutually exclusive. 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.

Skip to content 

FIELDS:

- `data_store_specs` (`list[genai.types.VertexAIStoreSpec] | None`)
- `datastore` (`str | None`)
- `engine` (`str | None`)
- `filter` (`str | None`)
- `max_results` (`int | None`)

field `data_store_specs`: `Optional[list[VertexAIStoreSpec]] = None` (alias '`dataStoreSpecs`')

Specifications that define the specific DataStores to be searched, along with configurations for those data stores. This is only considered for Engines with multiple data stores. It should only be set if engine is used.

field `datastore`: `Optional[str] = None`

Optional. Fully-qualified Vertex AI Search data store resource ID. Format:
`projects/{project}/locations/{location}/collections/{collection}/dataStores/{dataStore}`

field `engine`: `Optional[str] = None`

Optional. Fully-qualified Vertex AI Search engine resource ID. Format:
`projects/{project}/locations/{location}/collections/{collection}/engines/{engine}`

field `filter`: `Optional[str] = None`

Optional. Filter strings to be passed to the search API.

field `max_results`: `Optional[int] = None` (alias '`maxResults`')

Optional. Number of search results to return per query. The default value is 10. The maximum allowed value is 10.

pydantic model `genai.types.VertexAIStoreSpec`

Bases: `BaseModel`

Define data stores within engine to filter on in a search call and configurations for those data stores.

For more information, see <https://cloud.google.com/generative-ai-app-builder/docs/reference/rpc/google.cloud.discoveryengine.v1#datastorespec>

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

`PydanticValidationError][pydantic_core.ValidationError]` if the input data cannot be validated to Skip to content node.

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

► Show JSON schema

FIELDS:

- `data_store` (str | None)
- `filter` (str | None)

field `data_store`: Optional[str] = None (alias 'dataStore')

Full resource name of DataStore, such as Format:

`projects/{project}/locations/{location}/collections/{collection}/dataStores/{dataStore}`

field `filter`: Optional[str] = None

Optional. Filter specification to filter documents in the data store specified by `data_store` field. For more information on filtering, see [Filtering](<https://cloud.google.com/generative-ai-app-builder/docs/filter-search-metadata>)

class `genai.types.VertexAIStoreSpecDict`

Bases: `TypedDict`

Define data stores within engine to filter on in a search call and configurations for those data stores.

For more information, see <https://cloud.google.com/generative-ai-app-builder/docs/reference/rpc/google.cloud.discoveryengine.v1#datastorespec>

`data_store`: Optional[str]

`projects/{project}/locations/{location}/collections/{collection}/dataStores/{dataStore}`

TYPE:

Full resource name of DataStore, such as Format

`filter`: Optional[str]

`//cloud.google.com/generative-ai-app-builder/docs/filter-search-metadata`

TYPE:

Optional. Filter specification to filter documents in the data store specified by `data_store` field. For more information on filtering, see [Filtering](<https://cloud.google.com/generative-ai-app-builder/docs/filter-search-metadata>)

class `genai.types.VertexAISearchDict`

Bases: `TypedDict`

Retrieve from Vertex AI Search datastore or engine for grounding.

Skip to content

datastore and engine are mutually exclusive. See <https://cloud.google.com/products/agent-builder>

data_store_specs: `Optional [list [VertexAIStoreSpecDict]]`

Specifications that define the specific DataStores to be searched, along with configurations for those data stores. This is only considered for Engines with multiple data stores. It should only be set if engine is used.

datastore: `Optional [str]`

`projects/{project}/locations/{location}/collections/{collection}/dataStores/{dataStore}`

TYPE:

Optional. Fully-qualified Vertex AI Search data store resource ID. Format

engine: `Optional [str]`

`projects/{project}/locations/{location}/collections/{collection}/engines/{engine}`

TYPE:

Optional. Fully-qualified Vertex AI Search engine resource ID. Format

filter: `Optional [str]`

Optional. Filter strings to be passed to the search API.

max_results: `Optional [int]`

Optional. Number of search results to return per query. The default value is 10. The maximum allowed value is 10.

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

Skip to content

FIELDS:

- `rag_corpora` (`list[str] | None`)
- `rag_resources` (`list[genai.types.VertexRagStoreRagResource] | None`)
- `rag_retrieval_config` (`genai.types.RagRetrievalConfig | None`)
- `similarity_top_k` (`int | None`)
- `store_context` (`bool | 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 `rag_retrieval_config`: `Optional[RagRetrievalConfig] = None (alias 'ragRetrievalConfig')`

Optional. The retrieval config for the Rag query.

field `similarity_top_k`: `Optional[int] = None (alias 'similarityTopK')`

Optional. Number of top k results to return from the selected corpora.

field `store_context`: `Optional[bool] = None (alias 'storeContext')`

Optional. Currently only supported for Gemini Multimodal Live API. In Gemini Multimodal Live API, if `store_context` bool is specified, Gemini will leverage it to automatically memorize the interactions between the client and Gemini, and retrieve context when needed to augment the response generation for users' ongoing and future interactions.

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

Bases: `TypedDict`

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.

rag_retrieval_config: `Optional[RagRetrievalConfigDict]`

Optional. The retrieval config for the Rag query.

similarity_top_k: `Optional[int]`

Optional. Number of top k results to return from the selected corpora.

store_context: `Optional[bool]`

Optional. Currently only supported for Gemini Multimodal Live API. In Gemini Multimodal Live API, if `store_context` bool is specified, Gemini will leverage it to automatically memorize the interactions between the client and Gemini, and retrieve context when needed to augment the response generation for users' ongoing and future interactions.

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

Skip to content

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`

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`

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

Skip to content

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.

classmethod `from_file`(*, location, mime_type=None)

Loads a video from a local file.

RETURN TYPE:

`Video`

PARAMETERS:

- **location** – The local path to load the video from.
- **mime_type** – The MIME type of the video. If not provided, the MIME type will be automatically determined.

RETURNS:

A loaded video as an `Video` object.

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`.

[Skip to content](#)

This method only works in a notebook environment.

RETURN TYPE:

None

class genai.types.VideoCompressionQuality(*values)

Bases: CaseInsensitiveEnum

Enum that controls the compression quality of the generated videos.

LOSSLESS = 'LOSSLESS'

Lossless video compression quality. This will produce videos with a larger file size.

OPTIMIZED = 'OPTIMIZED'

Optimized video compression quality. This will produce videos with a compressed, smaller file size.

class genai.types.VideoDict

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

Describes how the video in the Part should be used by 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.

Skip to content

↓ schema

FIELDS:

- `end_offset (str | None)`
- `fps (float | None)`
- `start_offset (str | None)`

field `end_offset`: `Optional[str] = None (alias 'endOffset')`

Optional. The end offset of the video.

field `fps`: `Optional[float] = None`

The frame rate of the video sent to the model. If not specified, the default value will be 1.0.

The fps range is (0.0, 24.0].

field `start_offset`: `Optional[str] = None (alias 'startOffset')`

Optional. The start offset of the video.

class `genai.types.VideoMetadataDict`

Bases: `TypedDict`

Describes how the video in the Part should be used by the model.

`end_offset`: `Optional[str]`

Optional. The end offset of the video.

`fps`: `Optional[float]`

The frame rate of the video sent to the model. If not specified, the default value will be 1.0.

The fps range is (0.0, 24.0].

`start_offset`: `Optional[str]`

Optional. The start offset of the video.

pydantic model `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 implicitly positional-only to allow `self` as a field name.

Skip to content

`\N{schema symbol}`

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.

pydantic model `genai.types.WeightedPrompt`

Bases: `BaseModel`

Maps a prompt to a relative weight to steer music 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:

- `text` (`str` | `None`)
- `weight` (`float` | `None`)

field `text`: `Optional[str] = None`

Text prompt.

field `weight`: `Optional[float] = None`

Weight of the prompt. The weight is used to control the relative importance of the prompt. Higher weights are more important than lower weights.

Weight must not be 0. Weights of all `weighted_prompts` in this `LiveMusicClientContent` message will be normalized.

Skip to content

`genai.types.WeightedPromptDict`

Bases: `TypedDict`

Maps a prompt to a relative weight to steer music generation.

text: Optional [str]

Text prompt.

weight: Optional [float]

Weight of the prompt. The weight is used to control the relative importance of the prompt.

Higher weights are more important than lower weights.

Weight must not be 0. Weights of all weighted_prompts in this LiveMusicClientContent message will be normalized.

Copyright © 2024, Google

Made with Sphinx and @pradyunsg's Furo