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

Vertex AI SDK migration guide

The Generative AI module in the Vertex AI SDK is deprecated and will no longer be available after June 24, 2026. The Google Gen AI SDK contains all the capabilities of the Vertex AI SDK, and supports many additional capabilities.

Use this migration guide (#migration) to convert Python, Java, JavaScript, and Go code using the Vertex AI SDK to the Google Gen AI SDK.

Key changes

The following namespaces in the Vertex AI SDK are in the deprecation phase. SDK releases after June 24, 2026 won't include theses modules. Use the equivalent namespaces from the Google Gen AI SDK, which has full feature parity with the deprecated modules and packages.

Vertex AI SDK	Impacted code
`google-cloud-aiplatform` (https://github.com/googleapis/python-aiplatform)	Removed modules: • `vertexai.generative_models` (https://github.com/googleapis/python-aiplatform/tree/main/vertexai/generative_models) • `vertexai.language_models` (https://github.com/googleapis/python-aiplatform/tree/main/vertexai/language_models) • `vertexai.vision_models` (https://github.com/googleapis/python-aiplatform/tree/main/vertexai/vision_models) • `vertexai.caching` (https://github.com/googleapis/python-aiplatform/tree/main/vertexai/caching)

Vertex AI SDK	Impacted code
	• `vertexai.tuning` (https://github.com/googleapis/python-aiplatform/tree/main/vertexai/tuning)
`cloud.google.com/go/vertexai/genai` (http://cloud.google.com/go/vertexai/genai)	Removed package: • `vertex.genai` (https://pkg.go.dev/cloud.google.com/go/vertexai@v0.13.4/gena
`@google-cloud/vertexai` (https://github.com/googleapis/nodejs-vertexai)	Removed modules: • `vertexai.generative_models` (https://github.com/googleapis/nodejs-vertexai/blob/main/src/models/generative_models.ts) • `vertexai.chat_session` (https://github.com/googleapis/nodejs-vertexai/blob/main/src/models/chat_session.ts) • `vertexai.functions` (https://github.com/googleapis/nodejs-vertexai/tree/main/src/functions)
`com.google.cloud:google-cloud-vertexai` (https://github.com/googleapis/google-cloud-java/tree/main/java-vertexai)	Removed package: • `com.google.cloud.vertexai.generativeai` (https://github.com/googleapis/google-cloud-java/tree/main/java-vertexai/google-cloud-vertexai/src/main/java/com/google/cloud/vertexai/generativeai)

Code migration

Use the following sections to migrate specific code snippets from the Vertex AI SDK to the Google Gen AI SDK.

Note: The examples may omit imports, dependencies, and other boilerplate code to improve readability.

Installation

Replace the Vertex AI SDK dependency with the Google Gen AI SDK dependency.

Before

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

```
go get cloud.google.com/go/vertexai/genai
```

After

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

```
go get google.golang.org/genai
```

Context caching

Context caching involves storing and reusing frequently used portions of model prompts for similar requests. Replace the Vertex AI SDK implementation with the Google Gen AI SDK dependency.

Before

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

Imports

```
package contextcaching

// [START generativeaionvertexai_gemini_create_context_cache]
import (
  "context"
  "fmt"
  "io"
  "time"

  "cloud.google.com/go/vertexai/genai"
)
```

Create

```
content := &genai.CachedContent{
  Model: modelName,
  SystemInstruction: &genai.Content{
    Parts: []genai.Part{genai.Text(systemInstruction)},
  },
  Expiration: genai.ExpireTimeOrTTL{TTL: 60 * time.Minute},
  Contents: []*genai.Content{
    {
      Role:  "user",
      Parts: []genai.Part{part1, part2},
    },
  },
}

result, err := client.CreateCachedContent(context, content)
```

Get

```
cachedContent, err := client.GetCachedContent(context, contentName)
```

Delete

```
err = client.DeleteCachedContent(context, contentName)
```

Update

```
newExpireTime := cc.Expiration.ExpireTime.Add(15 * time.Minute)
ccUpdated := client.UpdateCachedContent(context, cc, &genai.CachedContentToUpda
        Expiration: &genai.ExpireTimeOrTTL{ExpireTime: newExpireTime},
})
```

List

```
iter, err := client.ListCachedContents(context, contentName)
```

After

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

Imports

```
import (
  "context"
  "encoding/json"
  "fmt"
  "io"

  genai "google.golang.org/genai"
)
```

Create

```
cacheContents := []*genai.Content{
  {
    Parts: []*genai.Part{
      {FileData: &genai.FileData{
        FileURI:  "gs://cloud-samples-data/generative-ai/pdf/2312.11805v3.pdf",
        MIMEType: "application/pdf",
      }},
      {FileData: &genai.FileData{
        FileURI:  "gs://cloud-samples-data/generative-ai/pdf/2403.05530.pdf",
        MIMEType: "application/pdf",
      }},
    },
    Role: "user",
  },
}
config := &genai.CreateCachedContentConfig{
  Contents: cacheContents,
  SystemInstruction: &genai.Content{
    Parts: []*genai.Part{
```

```
      {Text: systemInstruction},
    },
  },
  DisplayName: "example-cache",
  TTL:         "86400s",
}

res, err := client.Caches.Create(ctx, modelName, config)
```

Get

```
cachedContent, err := client.GetCachedContent(ctx, contentName)
```

Delete

```
_, err = client.Caches.Delete(ctx, result.Name, &genai.DeleteCachedContentConfi
```

Update

```
result, err = client.Caches.Update(ctx, result.Name, &genai.UpdateCachedContent
  ExpireTime: time.Now().Add(time.Hour),
})
```

List

```
// List the first page.
page, err := client.Caches.List(ctx, &genai.ListCachedContentsConfig{PageSize:

// Continue to the next page.
page, err = page.Next(ctx)

// Resume the page iteration using the next page token.
page, err = client.Caches.List(ctx, &genai.ListCachedContentsConfig{PageSize: 2
```

Configuration and system instructions

Configuration defines parameters that control model behavior, and system instructions provide guiding directives to steer model responses towards a specific persona, style, or task. Replace the configuration and system instructions from the Vertex AI SDK with the following code that uses the Google Gen AI SDK.

Before

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

```
import (
  "context"
  "cloud.google.com/go/vertexai/genai"
)

model := client.GenerativeModel(modelName)

model.GenerationConfig (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest/
  TopP:            proto.Float32(1),
  TopK:            proto.Int32(32),
  Temperature:     proto.Float32(0.4),
  MaxOutputTokens: proto.Int32(2048),
}

systemInstruction := fmt.Sprintf("Your mission is to translate text from %xs to

model.SystemInstruction = &genai.Content (https://cloud.google.com/vertex-ai/generative-ai/dc
  Role:  "user",
  Parts: []genai.Part (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest/g
}
```

After

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

```
import (
  "context"
  genai "google.golang.org/genai"
)
```

```
config := &genai.GenerateContentConfig{
  SystemInstruction: &genai.Content{
    Parts: []*genai.Part{
      {Text: "You're a language translator. Your mission is to translate text i
    },
  },
}

resp, err := client.Models.GenerateContent(ctx, modelName, contents, config)
```

Embeddings

Embeddings are numerical vector representations of text, images, or video that capture their semantic or visual meaning and relationships in a high-dimensional space. Replace the embedding implementation from the Vertex AI SDK with the following code that uses the Google Gen AI SDK.

Before

Python (#python) Java (#java) JavaScript (#javascript) Go (#go)

Embeddings are not supported by the Go Vertex AI SDK, but are supported by the Google Gen AI SDK.

After

Python (#python) Java (#java) JavaScript (#javascript) Go (#go)

```
import (
  "context"
  "fmt"
  "google.golang.org/genai"
)

result, err := client.Models.EmbedContent(ctx, *model, genai.Text("What is your
fmt.Printf("%#v\n", result.Embeddings[0])

fmt.Println("Embed content RETRIEVAL_DOCUMENT task type example.")
result, err = client.Models.EmbedContent(ctx, *model, genai.Text("What is your
fmt.Printf("%#v\n", result.Embeddings[0])
```

Function calling

Function calling enables a model to identify when to invoke an external tool or API and then generate structured data containing the necessary function and arguments for execution. Replace the function calling implementation with the Vertex AI SDK with the following code that uses the Google Gen AI SDK.

Before

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

```
package functioncalling

import (
  "context"
  "encoding/json"
  "errors"
  "fmt"
  "io"

  "cloud.google.com/go/vertexai/genai"
)

funcName := "getCurrentWeather"
funcDecl := &genai.FunctionDeclaration (https://cloud.google.com/vertex-ai/generative-ai/docs
  Name:        funcName,
  Description: "Get the current weather in a given location",
  Parameters: &genai.Schema (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/
    Type: genai.TypeObject (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/la
    Properties: map[string]*genai.Schema (https://cloud.google.com/vertex-ai/generative-ai/dc
      "location": {
        Type:        genai.TypeString (https://cloud.google.com/vertex-ai/generative-ai/docs/
        Description: "location",
      },
    },
    Required: []string{"location"},
  },
}

// Add the weather function to our model toolbox.
model.Tools = []*genai.Tool (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/
  {
    FunctionDeclarations: []*genai.FunctionDeclaration (https://cloud.google.com/vertex
  },
}
```

```
prompt := genai.Text (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest/ge
resp, err := model.GenerateContent (https://cloud.google.com/vertex-ai/generative-ai/docs/refe

if len(resp.Candidates) == 0 {
  return errors.New (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest/gen
} else if len(resp.Candidates[0].FunctionCalls (https://cloud.google.com/vertex-ai/generat
  return errors.New (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest/gen
}

funcResp := &genai.FunctionResponse (https://cloud.google.com/vertex-ai/generative-ai/docs/ref
  Name: funcName,
  Response: map[string]any{
    "content": mockAPIResp,
  },
}

// Return the API response to the model allowing it to complete its response.
resp, err = model.GenerateContent (https://cloud.google.com/vertex-ai/generative-ai/docs/refere
if err != nil {
  return fmt.Errorf("failed to generate content: %w", err)
}
if len(resp.Candidates) == 0 || len(resp.Candidates[0].Content (https://cloud.google.c
  return errors.New (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest/gen
}
```

After

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

```
package main

import (
  "context"
  "encoding/json"
  "flag"
  "fmt"
  "log"

  "google.golang.org/genai"
)

var model = flag.String("model", "gemini-2.5-flash", "the model name, e.g. gemi
```

```
func run(ctx context.Context) {
  client, err := genai.NewClient(ctx, nil)
  if err != nil {
    log.Fatal(err)
  }

  funcName := "getCurrentWeather"
  funcDecl := &genai.FunctionDeclaration{
    Name:        funcName,
    Description: "Get the current weather in a given location",
    Parameters: &genai.Schema{
      Type: genai.TypeObject,
      Properties: map[string]*genai.Schema{
        "location": {
          Type:        genai.TypeString,
          Description: "location",
        },
      },
      Required: []string{"location"},
    },
  }
  // Add the weather function to our model toolbox.
  var config *genai.GenerateContentConfig = &genai.GenerateContentConfig{
    Tools: []*genai.Tool{
      {
        FunctionDeclarations: []*genai.FunctionDeclaration{funcDecl},
      },
    },
  }
  // Call the GenerateContent method.
  result, err := client.Models.GenerateContent(ctx, *model, genai.Text("What's
  if err != nil {
    log.Fatal(err)
  }
  fmt.Println(result.Candidates[0].Content.Parts[0].FunctionCall.Name)

  // Use synthetic data to simulate a response from the external API.
  // In a real application, this would come from an actual weather API.
  mockAPIResp, err := json.Marshal(map[string]string{
    "location":         "Boston",
    "temperature":      "38",
    "temperature_unit": "F",
    "description":      "Cold and cloudy",
    "humidity":         "65",
    "wind":             `{"speed": "10", "direction": "NW"}`,
  })
  if err != nil {
    log.Fatal(err)
```

```
  }

  funcResp := &genai.FunctionResponse{
    Name: funcName,
    Response: map[string]any{
      "content": mockAPIResp,
    },
  }

  // Return the API response to the model allowing it to complete its response.
  mockedFunctionResponse := []*genai.Content{
    &genai.Content{
      Role: "user",
      Parts: []*genai.Part{
        &genai.Part{Text: "What's the weather like in Boston?"},
      },
    },
    result.Candidates[0].Content,
    &genai.Content{
      Role: "tool",
      Parts: []*genai.Part{
        &genai.Part{FunctionResponse: funcResp},
      },
    },
  }
  result, err = client.Models.GenerateContent(ctx, *model, mockedFunctionRespon
  if err != nil {
    log.Fatal(err)
  }
  fmt.Println(result.Text())
}

func main() {
  ctx := context.Background()
  flag.Parse()
  run(ctx)
}
```

Grounding

Grounding is the process of providing a model with external, domain-specific information to improve response accuracy, relevance, and consistency. Replace the grounding implementation with the Vertex AI SDK with the following code that uses the Google Gen AI SDK.

Before

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

Grounding is not supported by the Go Vertex AI SDK, but is supported by the Google Gen AI SDK.

After

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

```
package main

import (
  "context"
  "flag"
  "fmt"
  "log"

  "google.golang.org/genai"
)

var model = flag.String("model", "gemini-2.5-flash", "the model name, e.g. gemi

func run(ctx context.Context) {
  client, err := genai.NewClient(ctx, nil)
  if err != nil {
    log.Fatal(err)
  }

  // Add the Google Search grounding tool to the GenerateContentConfig.
  var config *genai.GenerateContentConfig = &genai.GenerateContentConfig{
    Tools: []*genai.Tool{
      {
        GoogleSearch: &genai.GoogleSearch{},
      },
    },
  }
  // Call the GenerateContent method.
  result, err := client.Models.GenerateContent(ctx, *model, genai.Text("Why is
  if err != nil {
    log.Fatal(err)
  }
  fmt.Println(result.Text())
}
```

```
func main() {
  ctx := context.Background()
  flag.Parse()
  run(ctx)
}
```

Safety settings

Safety settings are configurable parameters that allow users to manage model responses by filtering or blocking content related to specific harmful categories, such as hate speech, sexual content, or violence. Replace the safety settings implementation with the Vertex AI SDK with the following code that uses the Google Gen AI SDK.

Before

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

```
package safetysettings

import (
  "context"
  "fmt"
  "io"

  "cloud.google.com/go/vertexai/genai"
)

// generateContent generates text from prompt and configurations provided.
func generateContent(w io.Writer, projectID, location, modelName string) error
  // location := "us-central1"
  // model := "gemini-2.5-flash"
  ctx := context.Background()

  client, err := genai.NewClient (https://cloud.google.com/vertex-ai/generative-ai/docs/referen
  if err != nil {
    return err
  }
  defer client.Close (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest/ge

  model := client.GenerativeModel(modelName)
  model.SetTemperature (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest/
```

```
  // configure the safety settings thresholds
  model.SafetySettings = []*genai.SafetySetting (https://cloud.google.com/vertex-ai/genera
    {
      Category:  genai.HarmCategoryHarassment (https://cloud.google.com/vertex-ai/generativ
      Threshold: genai.HarmBlockLowAndAbove (https://cloud.google.com/vertex-ai/generative-a
    },
    {
      Category:  genai.HarmCategoryDangerousContent (https://cloud.google.com/vertex-ai/g
      Threshold: genai.HarmBlockLowAndAbove (https://cloud.google.com/vertex-ai/generative-a
    },
  }

  res, err := model.GenerateContent (https://cloud.google.com/vertex-ai/generative-ai/docs/ref
  if err != nil {
    return fmt.Errorf("unable to generate content: %v", err)
  }
  fmt.Fprintf(w, "generate-content response: %v\n", res.Candidates[0].Content (h

  fmt.Fprintf(w, "safety ratings:\n")
  for _, r := range res.Candidates[0].SafetyRatings {
    fmt.Fprintf(w, "\t%+v\n", r)
  }

  return nil
}
```

After

Python (#python) Java (#java) JavaScript (#javascript) Go (#go)

```
package main

import (
  "context"
  "flag"
  "fmt"
  "log"

  "google.golang.org/genai"
)

var model = flag.String("model", "gemini-2.5-flash", "the model name, e.g. gemi
```

```
func run(ctx context.Context) {
  client, err := genai.NewClient(ctx, nil)
  if err != nil {
    log.Fatal(err)
  }

  var safetySettings []*genai.SafetySetting = []*genai.SafetySetting{
    {
      Category:  genai.HarmCategoryHarassment,
      Threshold: genai.HarmBlockThresholdBlockMediumAndAbove,
    },
    {
      Category:  genai.HarmCategoryDangerousContent,
      Threshold: genai.HarmBlockThresholdBlockMediumAndAbove,
    },
  }
  var config *genai.GenerateContentConfig = &genai.GenerateContentConfig{
    SafetySettings: safetySettings,
  }
  // Call the GenerateContent method.
  result, err := client.Models.GenerateContent(ctx, *model, genai.Text("What is
  if err != nil {
    log.Fatal(err)
  }
  fmt.Println(result.Text())
}

func main() {
  ctx := context.Background()
  flag.Parse()
  run(ctx)
}
```

Chat sessions

Chat sessions are conversational interactions where the model maintains context over multiple turns by recalling previous messages and using them to inform current responses. Replace the implementation from the Vertex AI SDK with the following code that uses the Google Gen AI SDK.

Before

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

```
import (
  "context"
  "errors"
  "fmt"

  "cloud.google.com/go/vertexai/genai"
)

prompt := "Do you have the Pixel 8 Pro in stock?"
fmt.Fprintf(w, "Question: %s\n", prompt)
resp, err := chat.SendMessage (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/g
```

After

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

```
package main

import (
  "context"
  "flag"
  "fmt"
  "log"

  "google.golang.org/genai"
)

var model = flag.String("model", "gemini-2.5-flash", "the model name, e.g. gemi

var config *genai.GenerateContentConfig = &genai.GenerateContentConfig{Temperat

// Create a new Chat.
chat, err := client.Chats.Create(ctx, *model, config, nil)

// Send first chat message.
result, err := chat.SendMessage(ctx, genai.Part{Text: "What's the weather in Sa
if err != nil {
  log.Fatal(err)
}
fmt.Println(result.Text())

// Send second chat message.
```

```
result, err = chat.SendMessage(ctx, genai.Part{Text: "How about New York?"})
if err != nil {
  log.Fatal(err)
}
fmt.Println(result.Text())
```

Multimodal inputs

Multimodal inputs refers to the ability of a model to process and understand information from data types beyond text, such as images, audio, and video. Replace the implementation with the Vertex AI SDK with the following code that uses the Google Gen AI SDK.

Before

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

Images

```
import (
  "context"
  "encoding/json"
  "fmt"
  "io"

  "cloud.google.com/go/vertexai/genai"
)

img := genai.FileData (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest/g
  MIMEType: "image/jpeg",
  FileURI:  "gs://generativeai-downloads/images/scones.jpg",
}
prompt := genai.Text (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest/ge

resp, err := gemini.GenerateContent (https://cloud.google.com/vertex-ai/generative-ai/docs/ref
if err != nil {
  return fmt.Errorf("error generating content: %w", err)
}
```

Video

```
package multimodalvideoaudio

import (
  "context"
  "errors"
  "fmt"
  "io"
  "mime"
  "path/filepath"

  "cloud.google.com/go/vertexai/genai"
)

part := genai.FileData (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest/
  MIMEType: mime.TypeByExtension(filepath.Ext("pixel8.mp4")),
  FileURI:  "gs://cloud-samples-data/generative-ai/video/pixel8.mp4",
}

res, err := model.GenerateContent (https://cloud.google.com/vertex-ai/generative-ai/docs/refere
    Provide a description of the video.
    The description should also contain anything important which people say in
`))
```

After

Python (#python) Java (#java) JavaScript (#javascript) Go (#go)

Images

```
package main

import (
  "context"
  "encoding/json"
  "flag"
  "fmt"
  "log"

  "google.golang.org/genai"
)
```

```
config := &genai.GenerateContentConfig{}
config.ResponseModalities = []string{"IMAGE", "TEXT"}
// Call the GenerateContent method.
result, err := client.Models.GenerateContent(ctx, *model, genai.Text("Generate
if err != nil {
  log.Fatal(err)
}
```

Video and Audio

```
package multimodalvideoaudio

import (
  "context"
  "errors"
  "fmt"
  "io"
  "mime"
  "path/filepath"

  "cloud.google.com/go/vertexai/genai"
)

part := genai.FileData (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest/
  MIMEType: mime.TypeByExtension(filepath.Ext("pixel8.mp4")),
  FileURI:  "gs://cloud-samples-data/generative-ai/video/pixel8.mp4",
}

res, err := model.GenerateContent (https://cloud.google.com/vertex-ai/generative-ai/docs/refere
    Provide a description of the video.
    The description should also contain anything important which people say in
`))
```

Text generation

Text generation is the process by which a model produces human-like written content based on a given prompt. Replace the implementation with the Vertex AI SDK with the following code that uses the Google Gen AI SDK.

Synchronous generation

Before

Python (#python) Java (#java) JavaScript (#javascript) Go (#go)

```
gemini := client.GenerativeModel(modelName)
prompt := genai.Text(
  "What's a good name for a flower shop that specializes in selling bouquets of

resp, err := gemini.GenerateContent(ctx, prompt)
```

After

Python (#python) Java (#java) JavaScript (#javascript) Go (#go)

```
var config *genai.GenerateContentConfig = &genai.GenerateContentConfig{Temperat
// Call the GenerateContent method.
result, err := client.Models.GenerateContent(ctx, *model, genai.Text("What is y
```

Asynchronous generation

Before

Python (#python) Java (#java) JavaScript (#javascript) Go (#go)

Not applicable: Go manages concurrent tasks without asynchronous operations.

After

Python (#python) Java (#java) JavaScript (#javascript) Go (#go)

Not applicable: Go manages concurrent tasks without asynchronous operations.

Streaming

Before

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

```
package streamtextbasic

import (
  "context"
  "errors"
  "fmt"
  "io"

  "cloud.google.com/go/vertexai/genai"
  "google.golang.org/api/iterator"
)

model := client.GenerativeModel(modelName)

iter := model.GenerateContentStream (https://cloud.google.com/vertex-ai/generative-ai/docs/ref
  ctx,
  genai.Text (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest/genai.html#cl
)
for {
  resp, err := iter.Next()
  fmt.Fprint(w, "generated response: ")
  for _, c := range resp.Candidates {
    for _, p := range c.Content (https://cloud.google.com/vertex-ai/generative-ai/docs/referenc
      fmt.Fprintf(w, "%s ", p)
    }
  }
}
```

After

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

```
client, err := genai.NewClient(ctx, nil)
var config *genai.GenerateContentConfig = &genai.GenerateContentConfig{SystemIn
// Call the GenerateContent method.
for result, err := range client.Models.GenerateContentStream(ctx, *model, genai
  if err != nil {
```

```
    log.Fatal(err)
  }
  fmt.Print(result.Text())
}
```

Image generation

Image generation is the process by which a models creates images from textual descriptions or other input modalities. Replace the implementation with the Vertex AI SDK with the following code that uses the Google Gen AI SDK.

Before

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

Image generation is not supported by the Go Vertex AI SDK, but is supported by the Google Gen AI SDK.

After

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

```
import (
  "encoding/json"
  "google.golang.org/genai"
)

fmt.Println("Generate image example.")
response1, err := client.Models.GenerateImages(
  ctx, "imagen-3.0-generate-002",
  /*prompt=*/ "An umbrella in the foreground, and a rainy night sky in the back
  &genai.GenerateImagesConfig{
    IncludeRAIReason:       true,
    IncludeSafetyAttributes: true,
    OutputMIMEType:         "image/jpeg",
  },
)
```

Controlled generation

Controlled generation refers to the process of guiding model output to adhere to specific constraints, formats, styles, or attributes, rather than generating free-form text. Replace the implementation with the Vertex AI SDK with the following code that uses the Google Gen AI SDK.

Before

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

```
import (
  "context"
  "cloud.google.com/go/vertexai/genai"
)

model.GenerationConfig (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest/

// Build an OpenAPI schema, in memory
model.GenerationConfig (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest/
  Type: genai.TypeArray (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest
  Items: &genai.Schema (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest/
    Type: genai.TypeArray (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/lat
    Items: &genai.Schema (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/late
      Type: genai.TypeObject (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go
      Properties: map[string]*genai.Schema (https://cloud.google.com/vertex-ai/generative-ai/
        "object": {
          Type: genai.TypeString (https://cloud.google.com/vertex-ai/generative-ai/docs/referen
        },
      },
    },
  },
}

img1 := genai.FileData (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest/
  MIMEType: "image/jpeg",
  FileURI:  "gs://cloud-samples-data/generative-ai/image/office-desk.jpeg",
}

img2 := genai.FileData (https://cloud.google.com/vertex-ai/generative-ai/docs/reference/go/latest/
  MIMEType: "image/jpeg",
  FileURI:  "gs://cloud-samples-data/generative-ai/image/gardening-tools.jpeg",
}

prompt := "Generate a list of objects in the images."
```

```
res, err := model.GenerateContent (https://cloud.google.com/vertex-ai/generative-ai/docs/refere
```

After

Python (#python) Java (#java) JavaScript (#javascript) Go (#go)

```
import (
  "context"
  "encoding/json"

  genai "google.golang.org/genai"
)

cacheContents := []*genai.Content{
  {
    Parts: []*genai.Part{
      {FileData: &genai.FileData{
        FileURI:  "gs://cloud-samples-data/generative-ai/pdf/2312.11805v3.pdf",
        MIMEType: "application/pdf",
      }},
      {FileData: &genai.FileData{
        FileURI:  "gs://cloud-samples-data/generative-ai/pdf/2403.05530.pdf",
        MIMEType: "application/pdf",
      }},
    },
    Role: "user",
  },
}

config := &genai.CreateCachedContentConfig{
  Contents: cacheContents,
  SystemInstruction: &genai.Content{
    Parts: []*genai.Part{
      {Text: systemInstruction},
    },
  },
  DisplayName: "example-cache",
  TTL:         "86400s",
}

res, err := client.Caches.Create(ctx, modelName, config)
if err != nil {
  return "", fmt.Errorf("failed to create content cache: %w", err)
}
```

```
cachedContent, err := json.MarshalIndent(res, "", "  ")
if err != nil {
  return "", fmt.Errorf("failed to marshal cache info: %w", err)
}
```

Count tokens

Tokens are the fundamental units of text (letters, words, phrases) that models process, analyze, and generate. To count or compute tokens in a response, replace the implementation with the Vertex AI SDK with the following code that uses the Google Gen AI SDK.

Before

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

```
package tokencount

import (
  "context"
  "fmt"
  "cloud.google.com/go/vertexai/genai"
)

resp, err := model.CountTokens(ctx, prompt)

fmt.Fprintf(w, "Number of tokens for the prompt: %d\n", resp.TotalTokens)

resp2, err := model.GenerateContent (https://cloud.google.com/vertex-ai/generative-ai/docs/ref

fmt.Fprintf(w, "Number of tokens for the prompt: %d\n", resp2.UsageMetadata (http
fmt.Fprintf(w, "Number of tokens for the candidates: %d\n", resp2.UsageMetadata
fmt.Fprintf(w, "Total number of tokens: %d\n", resp2.UsageMetadata (https://cloud.go
```

After

Python (#python)Java (#java)JavaScript (#javascript)Go (#go)

```
import (
  "context"
  "flag"
  "fmt"
  "log"

  "google.golang.org/genai"
)

client, err := genai.NewClient(ctx, &genai.ClientConfig{Backend: genai.BackendV

fmt.Println("Count tokens example.")
countTokensResult, err := client.Models.CountTokens(ctx, *model, genai.Text("Wh

fmt.Println(countTokensResult.TotalTokens)
```


Last updated 2025-08-28 UTC.